
ClearControls®
ClearView SCADA

TABLE OF CONTENTS

INTRODUCTION

Thank you for your purchase. Over the last twelve years ClearView has evolved into the most feature-rich SCADA package in the world. Whether you have 1000 data points or 50,000 data points, ClearView increases your control while lowering your overall cost.

Let's face it – in today's high-speed, high-tech environment, access to data is power. With ClearView, collecting, analyzing, and reporting critical plant-floor information is now done easily and quickly across the enterprise.

This manual should have everything you need to get started on your next project. If you would like further information on some of the "open" aspects of ClearView, just follow the web links in each section. You'll find explanations of OPC, ODBC, ActiveX®, Visual Basic, DCOM, and many more.

All of us understand the trust you have placed in our team with your purchase of ClearView. We are proud that you have made this decision. We stand ready to help your success along.

ClearView Features

Open access to industrial data stored in off-the-shelf, **ODBC** standard databases such as Microsoft Access, SQL Server, and Oracle, reducing the time and effort needed for reporting and sharing information with other applications.

Easy to use software reduces programming effort, making implementations quick and straightforward. Screen changes can be made while the process remains running, increasing productivity and eliminating unnecessary downtime.

Client – server architecture, provides a flexible, network platform for distributed automation solutions and allows remote access while securing critical data.

Redundancy can be implemented to increase system availability.

Economical and simple licensing to meet customer budgets without surprising development or increased tag count license fees. One low price includes development, runtime, and unlimited tag count licenses.

Complete control software package includes reporting, trending, alarm e-email paging, statistical process control, and database backup utilities for implementing total automation solutions. A complete package simplifies ordering, budgeting and ensures you have all the tools necessary to deliver a complete solution.

Standards based software allows communication with virtually any brand of embedded controller and programmable logic controller (PLC) with OPC drivers. ActiveX ensures easy integration with other applications.

Security features provide network-wide, individualized access and control down to the specific object level, creating fine balance between system security and access.

Run-Time Localization provides multilingual support for run-time SCADA environment.

System Requirements

To run ClearView, we recommend the following hardware and software:

Operating System

Workstations:

- Windows 10 Professional
- Windows 8, 8.1 Professional
- Windows 7 Professional & Embedded

Servers:

- Windows Server 2008, 2012, 2012-R2, 2013 and 2016

Hardware

ClearView Server minimum requirements

- Processor i5.
- 4GB RAM (multiple clients and over 1000 tags).
- 120 GB hard drive disk space.
- TCP/IP network interface adapter.
- Second TCP/IP network interface adapter for fast network communication (recommended for Redundant Server).

ClearView Client minimum requirements

- Processor i5.
- 4GB RAM (multiple clients and over 1000 tags).
- 120 GB hard drive disk space.
- TCP/IP network interface adapter.

Run ClearView-SCADA as User or Power User

Requirements to Run ClearView-SCADA as a Windows User or Power User:

If you plan to run ClearView not as an Administrator but as a User or Power User please follow the steps below.

1. Login as Administrator.
2. Open file explorer window, select Tools and then Folder Options. Go to the View tab and uncheck “Use simple file sharing (Recommended)”. Click Apply and OK then close the file explorer window. This exposes the Security tab in the File and Folder Permissions windows.
3. Install the ClearView-SCADA.
4. Open registry (Click Start, Run and type regedit)

5. Locate [HKEY_LOCAL_MACHINE\SOFTWARE\ClearControls] folder and change the permissions. Right click on the on the ClearControls in the tree view and set the permissions as below. **Users** and **PowerUsers** require Full Control
6. Set Full control for the Users and Power Users for the files Clearview.exe and CVServer.exe located in "C:\Program Files\ClearControls\ClearView" folder.
7. CLEARVIEW-SCADA directories where the ClearView-SCADA database, ClearView-SCADA .cvp and .ccp project files are located require full control for the **Users** and **PowerUsers**.

Registering ClearView

When you first start ClearView Client or ClearView Server, you can evaluate the product for 14 days. To continue using the product after the evaluation period, you must purchase a license.

To register ClearView Client and ClearView Server software, perform the following steps:

1. Start ClearView Client/Server.
2. Open Registration interface.
    a. For ClearView Server:
        i. Double-click on ClearView Server Icon on the task bar. ClearView Server interface will open.

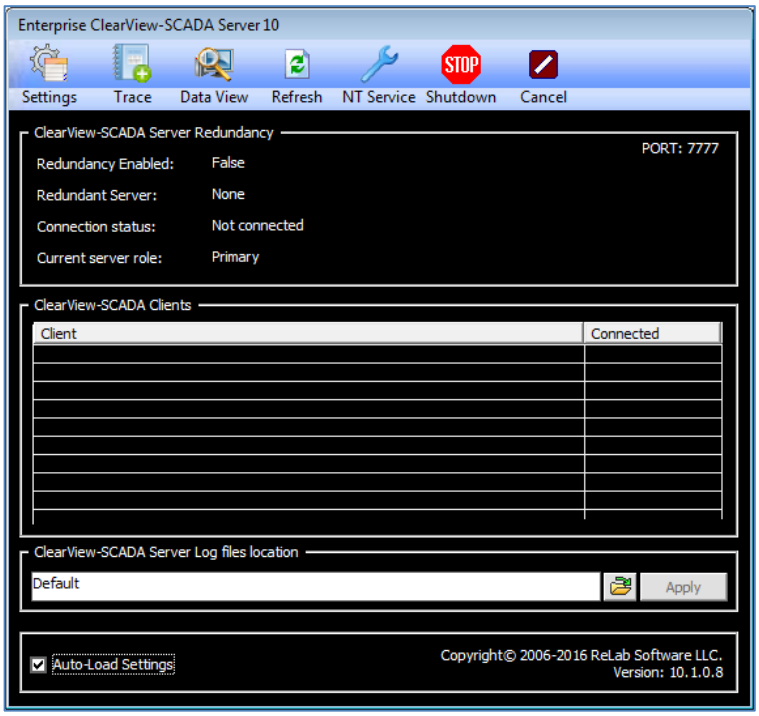


Figure #1

ii. Click **Settings**

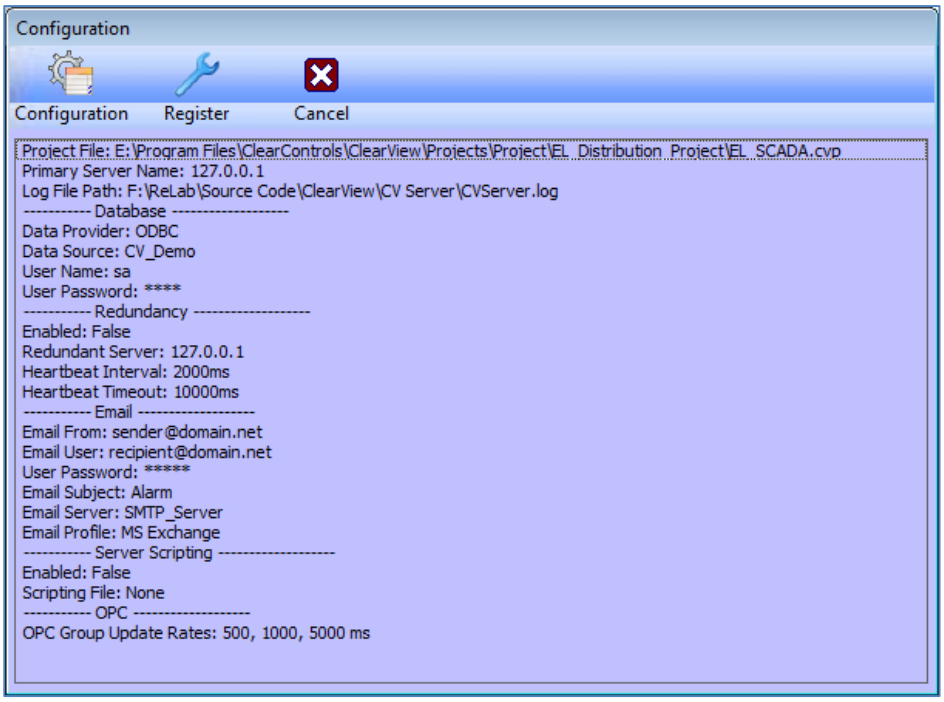


Figure #2

iii. Click **Register**

b. For ClearView Client:

i. In main menu click Help->Activate ClearView

3. ClearView Client (Figure #3) or ClearView Server (Figure #4) registration window will open. Chose Client or Server option depending on the license you purchased. Click **Register** button. The **Customer ID** number is displayed.
4. Contact ReLab Software. Provide the **Customer ID** number.
5. Enter the **Key Code** provided by ReLab Software.
6. Click **Register.**

Note: Once the software is registered, the Register window will not appear. Each PC that the software will be used on requires a purchased license.

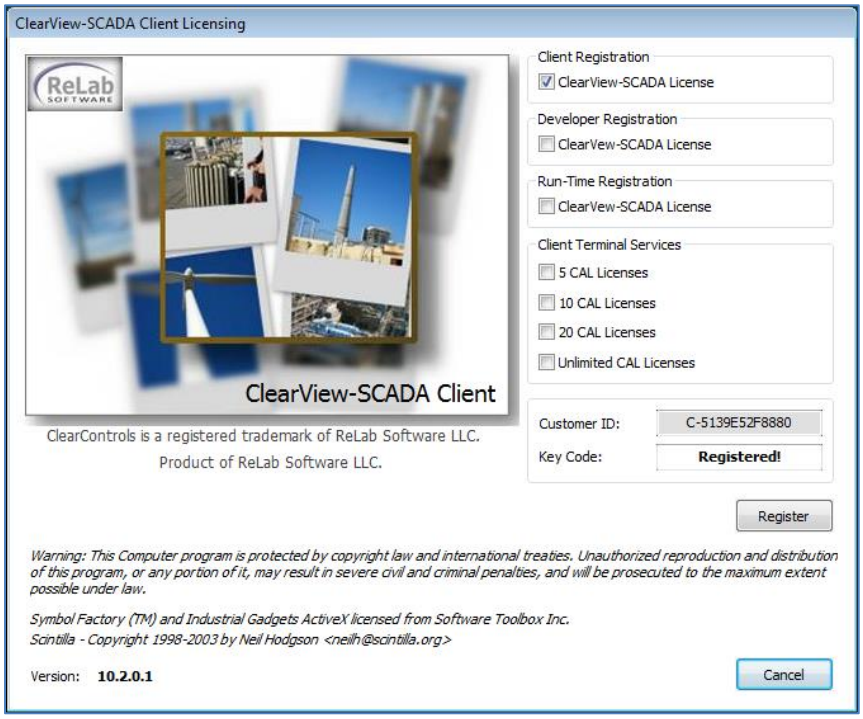


Figure #3

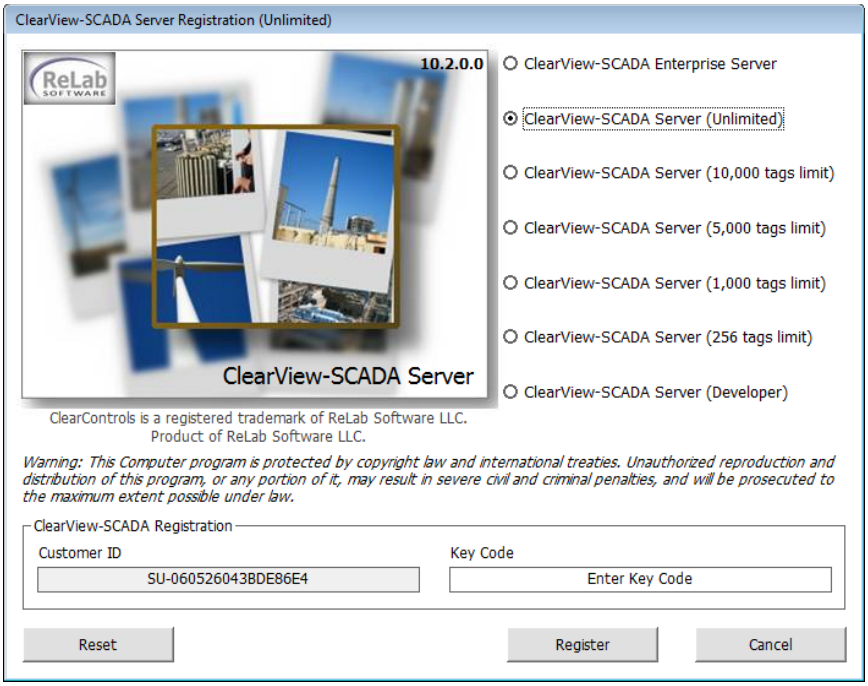


Figure #4

Warning: Pressing the “Reset” would reset the current ClearView-SCADA server Customer ID. The new license Key Code would be required. The “Reset” is only required when switching between ClearView-SCADA Server (256 tags limit, 1000 tags limit, 5000 tags limit, 10000 tags limit and Unlimited tags), and ClearView-SCADA Enterprise Server.

Running ClearView for the First Time

Follow the steps shown in the checklist below to configure ClearView after you launch the product for the very first time.

STEP	STEP DESCRIPTION	COMPLETE
1	Configure your OPC Server	
2	Launch ClearView client.	
3	Create a new project.	
4	Create database and tables in a database server.	
5	Configure user(s) and security group(s).	
6	Configure tag(s).	
7	Configure the alarm(s).	
8	Enable the alarm group(s) for this client computer.	
9	Specify trend group(s) if applicable.	
10	Create custom report(s) if applicable.	
11	Configure first ClearView screen.	
12	Add object(s) to the screen.	
13	Animate object(s) on the screen.	
14	Write a server script if applicable.	
15	Modify project settings if applicable.	
16	Save the project.	
17	Start and configure ClearView Server.	
18	Test your project.	

Technical Support

ReLab Software LLC.

2401 Stanwell Drive, Suite 300

Concord, CA 94520

MAIN: 925-262-4244

FAX: 925-262-4245

support@relabsoft.com

www.relabsoft.com

SECTION 1

ClearView Environment

ClearView combines Run and Design modes into a single application interface. On creation of a new project, ClearView displays the first screen in Design mode. Once a project is saved and reopened, all screens open in Run mode. Thereafter, security settings control the ability of the logged-in user to switch between Run and Design modes. The security settings also control what elements of ClearView environment are displayed and/or accessible.

This chapter discusses the key components of the ClearView environment.

The following figure illustrates the key elements of a ClearView screen in Design mode. When ClearView is switched to Run mode, the run-time Screen replaces the Drawing Pad and hides the Object Library and Properties windows. Each screen can be individually switched between Design and Run modes.

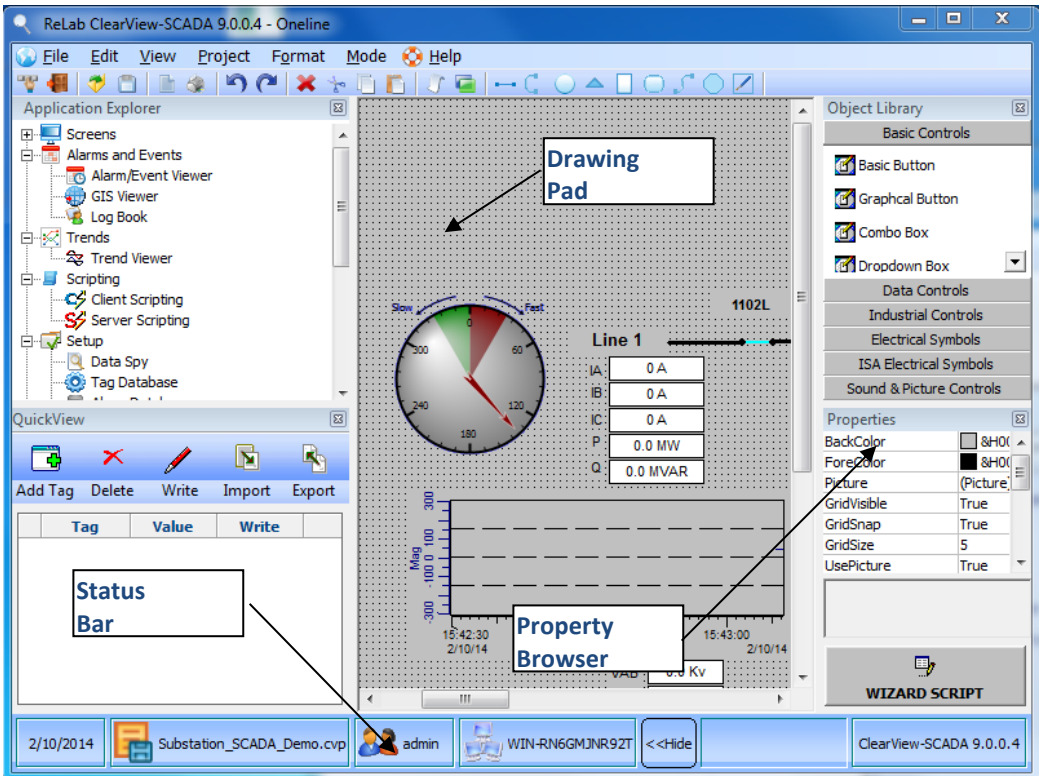


Figure # 1.1

Application Explorer – An organized dockable window that can access the complete project configuration

Menus – Access commands available

Toolbar – Provides easy access to common menu commands

Drawing Pad – A container for holding ActiveX controls and interacting with ClearView Server. When you switch to Run mode, the Drawing Pad becomes your screen interface. You can drag and drop graphics from the Object Library to the Drawing Pad.

- **Object Library** – A dockable window that accesses a library of ActiveX controls
- **Properties** – A dockable window that displays the properties of the selected control

- **ActiveX Objects** – Any graphical, numeric, text, drawing tool built on ActiveX technology
- **Status Bar** – Displays both alarm status and general project information
- **Wizard Script** – Accesses the application's scripting capability
- **Quick View –** Tag database access

Dockable Windows

Dockable windows are windows that can be moved or resized within the ClearView workspace. There are three dockable windows: The Application Explorer, the Object Library, and Properties.

To dock/undock a window, double click its title bar. You can also click and drag the window.

Note: If the Properties or Object Library windows becomes inaccessible because of moving/resizing the window beyond window boundaries, switch to Run mode, then switch back to Design mode.

If the Application Explorer becomes inaccessible, save the project; then exit and reopen ClearView.

Application Explorer

ClearView Application Explorer uses a hierarchal representation to display ClearView project configuration tools and screens. It shows the screens created in the project and gives quick access to database configuration tools.

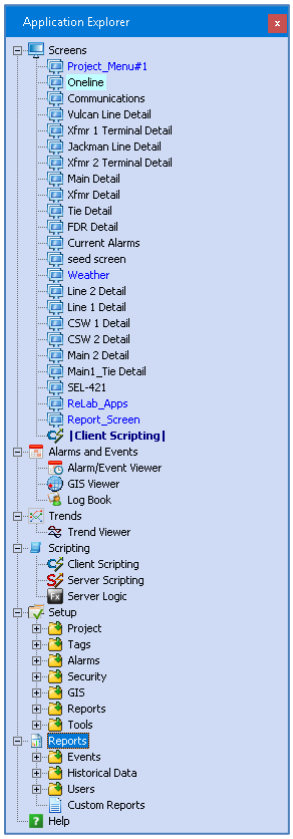


Figure # 1.2

To hide/show Application Explorer:

1. On the **View** menu, click **App Explorer**. (By default, the Application Explorer will be visible after creating a new project,)

2. Repeat step 1 to close it.

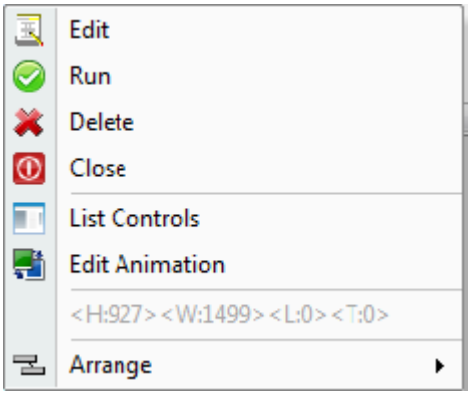
Note: Settings in **Security Groups** can be set to hide/unhide the Application Explorer depending on the security classification of the user who is logged in.

To navigate Application Explorer:

Double-click the folder or click the [+] sign to expand and view folder contents.

Double-click the folder or click the [-] sign to collapse the folder.

The following table briefly describes each item in ClearView **Application Explorer** with possible actions and results

Item	Description
Screens	Screen Name Double Click the screen name to Open/Show screen. Note: To edit the screen properties, right-click the screen name to see a pop-up menu: 
	Edit – Displays “Edit Screen” dialog box.
	Run/Design – Switches between Run and Design modes.
	Delete – Deletes the selected screen.
	Close – Closes the selected screen.
	List Controls – Lists the created controls.
	Edit Animation – Opens Animation Dialog, see SECTION 22 below.
	Arrange – Arranges screen order.
Alarms and Events	Alarm and Events Viewer

Item	Description
	Opens Alarm & Event/Event Viewer Interface. GIS Viewer Opens the GIS Viewer, see SECTION 13 below. Log Book Opens Log Book Interface.
Trends	Trend Viewer Opens Trend Viewer Interface.
Scripting	Client Scripting Opens Client Scripting development interface. Server Scripting Opens Server Scripting development interface.

Item	Description
Setup	Data Spy Executes Data Spy command to open Data Spy. Allows you to see all or a filtered list of current tag values. Tag Database Executes Tag Database command to open the Tag Editor to Add, Delete, or Edit tags. Alarm Database Executes the Alarm Database command to open the Alarm Tag Editor to add, delete, and edit tags. Alarm Groups Executes the Alarm Groups command on the Project\Setup menu to open the Alarm Group dialog box, allowing you to select which alarm groups will be active on the current ClearView Client station. Note: ClearView Server may require reboot after removing an Alarm Group. SQL Builder Executes the SQL Builder command on the Project\Setup menu to opens the SQL Builder dialog box. Users Database Executes the Users Database command to open the Users window, which provides ability to add/edit/delete users and edit Group Security. Object Toolbar Editor Executes the Object Toolbar Editor command to open Object Toolbar Editor dialog box. Backup Executes the Backup Command to open the File Backup dialog box, which allows you to select files to backup in *.ZIP format. Note: The Backup command on the File menu provides a backup of the database and selected files Report Editor Opens ClearView Reports Configuration Interface (Reportizer), see SECTION 12.

Item	Description
Reports	Several Built-in Crystal Reports are available. Right-click a report name to display the Report popup menu. Note: The Reports folder has links to display historical logs for Alarms, Events, Audited Activities, tag Data, and Security access. A Users Report and an option to open any external report is also available

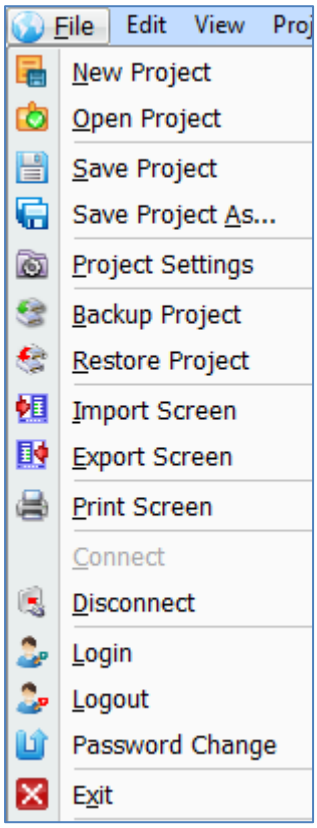
Main Menus

This section describes the different commands available under the different menus. When the current ClearView screen is in Run mode, the menu options for the user are different than when the screen is in Design mode. The Edit and Format menus are hidden when the top screen is in Run mode. Mode control is accessed under the Mode menu. Each screen has its own mode setting.

Security can be configured to allow or deny access to various commands on the menus.

File Menu

The table below describes the commands available on the File menu.

Menu	Menu Item	Description
File Menu  	New Project	Closes current project and opens Windows Open File dialog box. If the current project has been modified, a dialog box to save the project will display before exiting.
	Open Project	Closes current project and opens Window's Open File dialog box. If the current project has been modified, a dialog box to save the project will display before exiting.
	Save Project	Saves project. The location of the saved file is specified in the Project Settings window.
	Save Project As…	Opens Windows Save As dialog box to save the current project with a different name.
	Project Settings	Opens the Project Settings dialog box.
	Backup Project	Opens the Backup Project dialog box to create a file that has all the project information in a compressed format (*.zip).
	Restore Project	Opens the Restore Project dialog box to recreate a project from a compressed file (*.zip).
	Import Screen	Displays Window's Open File dialog box, allowing you to browse, select, and import a ClearView screen file (*.css). The imported screen is appended at the end of the screen list.
	Export Screen	Displays Window's Save File dialog box, allowing you to save the current screen in ClearView screen file (*.css) format.
	Print Screen	Opens the Print Screen preview window, displaying a preview of what the screen image will look like when it is printed.
	Connect	The Connect command makes the client reconnect to the server.
	Disconnect	The Disconnect command disconnects the ClearView Client from the ClearView server.
	Login/Logout	The Login command on the File menu opens the Login window, allowing you to enter a username and password to access a project. The Logout command logs out the current user. As a default, a logged out user is not allowed to modify screens, project settings, database table entries, or close the application. A project can also be set up to Auto Logout a user after a

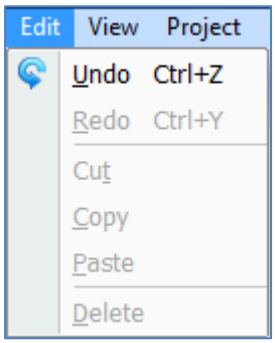
		specified time. Security settings define what access a logged in or logged out user has. **WARNING**: Always provide a way to log back in. If you logout manually or the system automatically logs you out without saving the modified screen, then you will lose your edits. ***Shortcuts: [Shift] + [L] – Login, [Shift] + [U] - Logout***
	Password Change	Opens a dialog box that allows the user can change their login password.
	Exit	Closes the ClearView application. If the project has been modified, a dialog box to save the project will display before exiting.

Edit Menu

The Edit menu commands only apply to screen edits. While the ClearView Client program is in Run mode, the Edit menu is disabled.

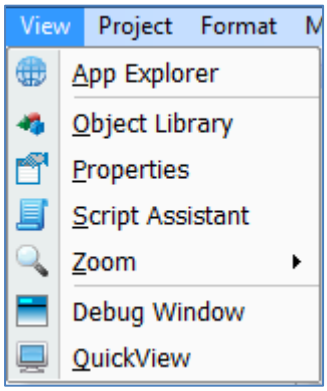
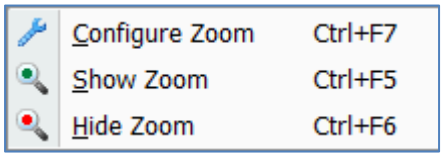
Note: The Undo and Redo edit commands do NOT apply to properties within objects.

The table below describes the commands available on the Edit menu.

Menu	Menu Item	Description
Edit Menu  	Undo	Changes the screen as if the last action you preformed had never happened. This command can be used after a deletion or undesired change has been made.
	Redo	Restores the changes to the screen performed by the Undo command. This command can be used to compare major changes in the project.
	Cut	Deletes an item or object from the screen while storing a copy to the Clipboard.
	Copy	Stores a copy of an item or object to the clipboard.
	Paste	Copies the contents of the Clipboard, if compatible, to the screen.
	Delete	Discards or removes an item or object from the screen.

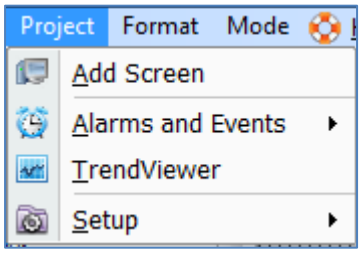
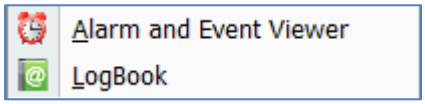
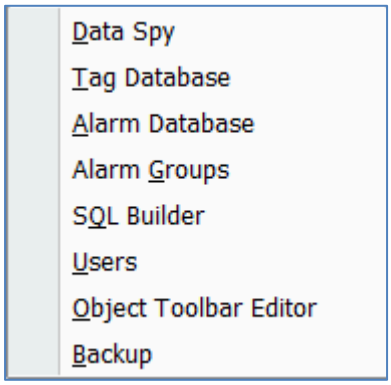
View Menu

The table below describes the commands available on the View menu.

Menu	Menu Item	Description
View menu 	App Explorer	Shows/Hides the Application Explorer window. The Application Explorer has links to most parts of a project and can access all screens.
	Object Library	Opens/Hides the Object Library window, which contains a library of ActiveX objects that can be added to a screen. Objects cannot be drawn onto a screen while in Run mode. This window is only available in Design mode.
	Properties	Opens/Hides the Properties window, which displays information about a selected object on the Screen. This is only available in Design mode.
	Script Assistant	Opens/Hides the Script Assistant window, which helps build references to ClearView scripts.
	Zoom	Shows Zoom sub-menu
	Debug Window	Shows/Hides the Debug Window, which displays outputs from the Debug.Print command. Outputs only visible if enabled.
	Quick View	Shows/Hides the run-time Quick View window, which allows selecting tags to display and write values to tags.
Zoom sub-menu 	Configure Zoom	Shows Zoom Setup interface
	Show Zoom	Shows the Zoom Box window, which lets a user see an enlarged section of the screen. The Zoom Box is centered on the section of the display that the mouse occupies
	Hide Zoom	Hides the Zoom Box window

Project Menu

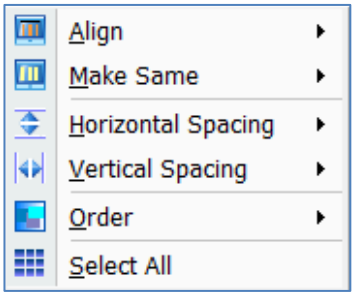
The table below describes the commands available on the Project menu.

Menu	Menu Item	Description
Project menu 	Add Screen	Opens the Create Screen window that allows a user to specify the dimensions of a new screen. The new screen is then added to the Screens folder of the Application Explorer window submenu
	Alarms & Events	Shows Alarms & Events sub-menu
	TrendViewer	Displays the Trend Viewer window, which allows a user to see the values of tags and how they change over time in different ranges and scales Setup submenu.
	Setup	Shows Setup sub-menu
Alarms & Events sub-menu 	Alarm and Event Viewer	Displays the Alarm/Event View window, which displays the information View of the alarms and events that can happen in the project.
	LogBook	Displays the Log Book window that lets users record the events that happen in the project. The Log Book is described in SECTION 4 below.
Setup sub-menu 	Data Spy	Displays the Data Spy window.
	Tag Database	Displays up the Tag Database window.
	Alarm Database	Displays the Alarm Database window.
	Alarm Groups	Displays the Alarm Groups window.
	SQL Builder	Displays the SQL Builder window.
	Users	Displays the Users window
	Object ToolBar	Displays the Object ToolBar editor.
	Backup	Displays the Backup window.

Format Menu

The commands on the Format menu are only available while a screen is in Design mode.

Note: The "Align" and "Make Same" format commands organize the object in reference to the last object selected in the group selection.

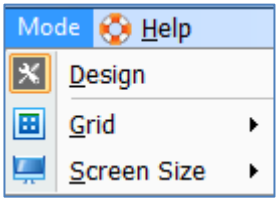
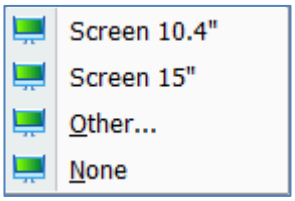
Menu	Menu Item	Description
Format menu  	Align	***Align Left*** makes a selected group of objects on the screen move to the left-most coordinate of the group. ***Align Right*** makes a selected group of objects on the screen move to the right-most coordinate of the group. ***Align Center*** makes a selected group of objects on the screen move to the center-most coordinate of the group. ***Align Top*** makes a selected group of objects on the screen move to the top-most coordinate of the group. ***Align Bottom*** makes a selected group of objects on the screen move to the bottom-most coordinate of the group. ***Align Middle*** makes a selected group of objects on the screen move to the middle-most coordinate of the group. ***Align to Grid*** makes a selected group of objects on the screen move to the closed-grid coordinate for each object.
	Make Same	***Height*** changes a selected group of objects to the same height. ***Width*** changes a selected group of objects to the same width. Both change the selected group of objects to the same height and width.
	Horizontal Spacing	***Make Equal*** changes a selected group of objects to the same horizontal distance away from each other. ***Increase*** changes a selected group of objects to have more horizontal distance away from each other. ***Decrease*** changes a selected group of objects to have less horizontal distance away from each other.
	Vertical Spacing	***Make Equal*** changes a selected group of objects to the same vertical distance away from each other. ***Increase*** changes a selected group of objects to have more vertical distance away from each other. ***Decrease*** changes a selected group of objects to have less vertical distance away from each other.
	Order	Controls overlapping or layering of controls. ***Send to Front*** makes the selected objects appear to be on top of all other objects. ***Send to Back*** makes the selected objects appear to be below all other objects. ***Bring Forward*** makes the selected objects appear to be one level upwards of another object. *Send Back* makes the selected object appear to be one layer below another object.
	Select All	Selects all objects on the screen

Mode Menu

Note: The commands on the Mode menu do not apply globally to all screens; they are used to change individual screen properties.

The **Grid** and **Screen size** have no effect when the project is in Run mode.

The table below describes the commands available on the Mode menu.

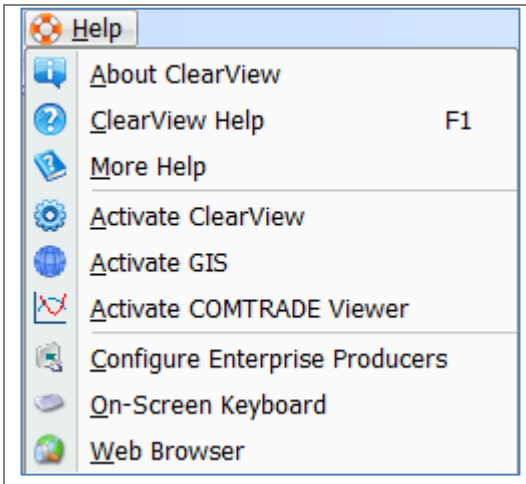
Menu	Menu Item	Description
Mode menu  	Design	Toggles the mode of the screen between design and run-time mode. If there is no checkmark on the left side of Design, then the topmost screen is in Run mode. Each screen has its own mode setting.
	Grid	
Grid sub-menu  	Show Grid (Sub menu)	Shows/Hides the foreground grid consisting of the pixel point of intersecting equal distance vertical and horizontal lines.
	Snap to Grid	Toggles Snap to Grid on or off. Forces object placement to the nearest grid point.
	Grid Size	Displays an entry form for specifying how many pixels between the grid lines.
Screen Size sub-menu  	Screen 10.4”	Sets screen size to 10.4”.
	Screen 15”	Sets screen size to 15”.
	Other	Opens Screen Width dialog.
	None	Sets full screen size mode.

Note: The screen Foreground Color property controls the grid color.

Help Menu

The table below describes the commands available on the Help menu.

Menu	Menu Item	Description
Help menu	About ClearView	Displays the ClearView about screen.
	ClearView Help	Displays the Help manual.

	More Help	Displays a list of help documents that have more detailed descriptions of the ClearView program.
	Activate ClearView	Opens ClearView Client activation screen.
	Activate GIS	Opens GIS activation screen.
	Activate COMTRADE Viewer	Opens COMTRADE Viewer activation screen.
	Configure Enterprise Producers	Opens Enterprise Producers Configuration Console.
	On-Screen Keyboard	Shows on-screen keyboard.
	Web Browser	Opens Web Browser screen.

Toolbar

Figure #1.3

The tools on ClearView main toolbar provide easy access to common ClearView tasks. The tools are organized and grouped according to similar functions. A ToolTip displays the tools name when the mouse hovers over it.

The following table briefly describes each tool:

	Executes the **Login** command on the File menu to open the Login window. *Shortcuts: [Shift] + [L] – Login*
	Executes the **Logout** command on the File menu to Logout the current user. *Shortcuts: [Shift] + [U] - Logout*
	Executes the **Open Project** command on the File menu to display the Open Project dialog box, allowing you to browse for existing projects.

	Executes the **Save Project** command on the File menu to save the current project to an already specified location.
	Executes the **Add Screen** command on the Project menu to display the Create Screen window.
	Executes the **Print Screen** command on the File menu to open the Print Screen window.
	Executes the **Undo** command on the Edit menu to reverse the last screen edit. ClearView retains the last 100 object edits.
	Executes the **Redo** command on the Edit menu to reverse the last Undo command.
	Executes the **Delete** command on the Edit menu to delete the selected object or group of objects.
	Executes the **Cut the Object** command on the Edit menu to delete the object from the screen and store it on the Clipboard.
	Executes the **Copy the Object** command on the Edit menu to copy the selected object or groups of objects to the Clipboard
	Executes the **Paste the Object** command on the Edit menu to paste the contents from the Clipboard to the screen, if compatible
	Executes the **Script Assistant** command on the File menu to show/hide the Script Assistant widow.

	Displays the Dynamo Object Library.

Note: The Dynamo Object Library is not accessible under the main menus.

Status Bar

The Status Bar at the bottom of the screen displays the overall status of the project running on the ClearView Client station.

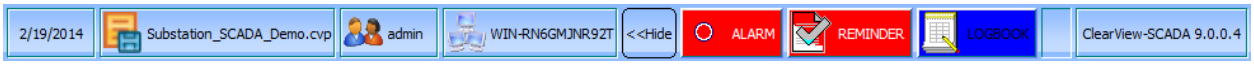
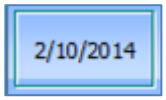
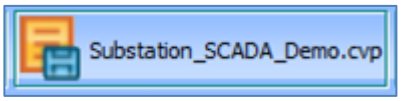
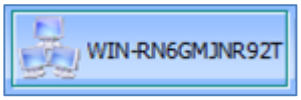


Figure # 1.4

Note: Settings in **Security Groups** can be set to hide/unhide the Status Bar depending on the security classification of each user who is logged in.

The following table describes the items contained in the Status Bar.

Item	Description
	Displays current system date (m/d/y).
	Displays project filename (*.cvp).
	Displays currently logged-in username.
	Displays status of connection to ClearView server.
	Hides the Status Bar items shown to the left of the Hide button.
	Log Book reminder.

	Button shown when any Alarm condition exists. A red background indicates an active, unacknowledged alarm. A yellow background indicates an inactive, unacknowledged alarm. Click **ALARM** to open the Alarm Viewer.
	A Reset QA button displays if an OPC server returns a bad quality on any tag. Clicking the button will open the Data Spy, filtered to display all bad quality tags detected.
	Appears when alarm is active. Opens Log Book Entry interface

Drawing Pad (Screen)

Each screen has its own Drawing Pad. A screen's Drawing Pad is available when the screen is placed in Design mode. The Drawing Pad is an ActiveX container, which maintains the properties and links to the ActiveX objects that are placed in it. The Drawing Pad has properties as well, accessible by clicking directly on the pad.

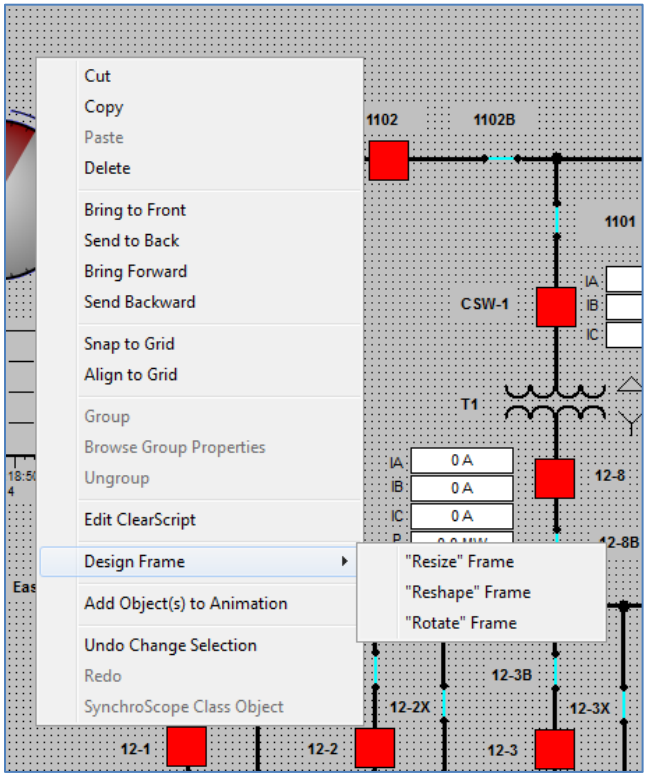


Figure # 1.5

ActiveX Objects

An ActiveX control is an object with properties, functions, and methods that allow interaction with ClearView Server and the OPC Servers. ActiveX controls are programmed in various programming languages and compiled into an *.ocx or *.dll object file.

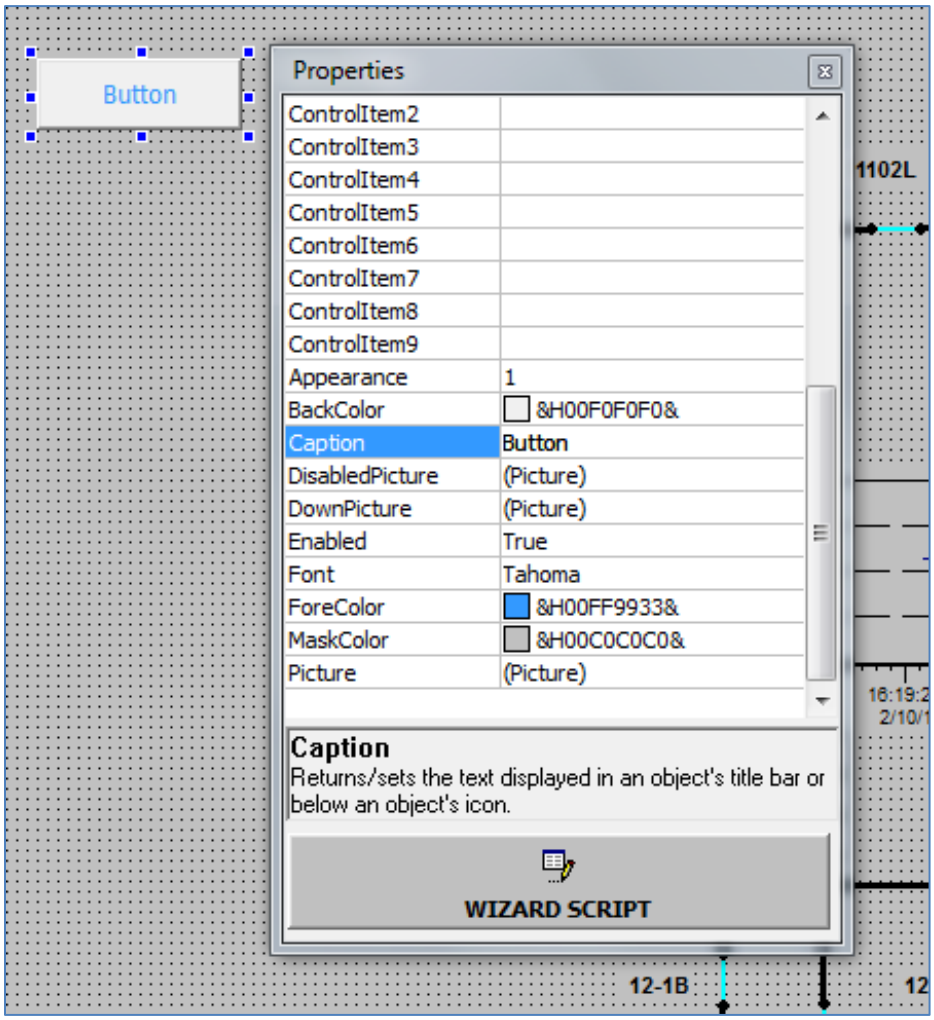


Figure # 1.6

ClearView Screens are built totally with ActiveX objects. The objects are added to the Drawing Pads of each screen while in Design mode. Once the screen is switched to Run mode, the ActiveX Controls become active.

Custom third-party controls can be added to ClearView library environment using the Object Tool Editor, accessible from the Application Explorer or from the Project > Settings menu.

For information on ActiveX Technology see Appendix A.

Object Library Window

The Object Library is a dockable window containing categorized lists of ActiveX objects that have been added to ClearView library. Objects can be added to or deleted from the Object Library using the **Object Toolbar Editor**, accessible from the Application Explorer or from the Project > Setup menu. The Object Library is only visible when the current screen is in Design mode.

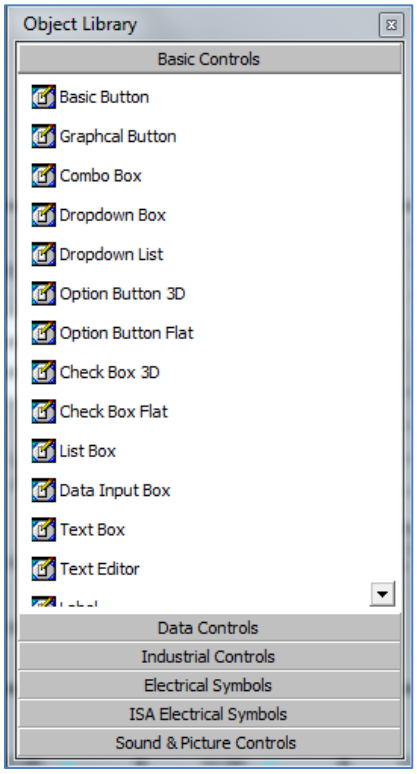


Figure # 1.7

Properties Window

The Properties window is a dockable window. The Properties window displays the list of properties for the selected ActiveX object. Common properties shared among a selected group can also be viewed.

The Properties window is only visible when the current screen is in Design mode.

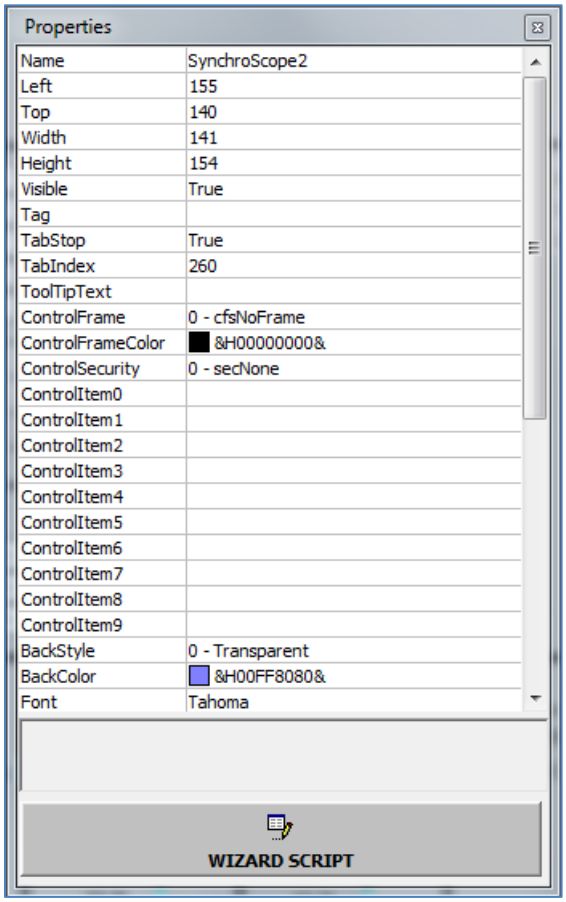


Figure # 1.8

Note: Many ActiveX objects provide their own custom property dialog boxes. If available, use the custom properties interface to configure the object.

SECTION 2

Project Setup

Both the ClearView Client and ClearView Server rely on the ClearView project file (*.cvp). This file is a text file containing the configuration information defining the following settings.

Note that Tags and Alarms for a particular project are stored in the project data base. Both Client and Server are using the data base source and credentials specified in the **[ClientDB]** section of the project's **.cvp** file. There is a possibility to have Client and Server to work with different databases, meaning that Client and Server are working with different sets of Tags of Alarms. For more information visit chapter **"More than one database per a setup"** at the end of this section.

[ClientMisc]	**[ServerRedundancy]**
Modified=11/21/2013 3:23:57 PM	RedundancyEnabled=0
UpdateCurrScreenOnly=1	ServerName=MIKES
EnableAutoLogin=1	**[ClientDB]**
Autousername=admin	DataProvider=ODBC
EnableGIS=0	DataSource=ClearControls
AutoLogout=0	DataSourceBack=
AutoLogoutInterval=15	DBInterval=0
AutoLogoutUser=	OraUsername=User Name
EnablePsuedoPassword=0	OraPassword=Password
AnimationInterval=0	OraUsernameBack=
OPCTagInterval=500	OraPasswordBack=
OPCTagInterval1=1000	DBPurgeThreshold=
OPCTagInterval2=5000	**[ClientUserInformation]**
Group=Administrator	Login=admin
QueueSize=10	Password=
AutoSaveIntervalMinutes=0	**[EventViewer]**
[ClientServer]	AlarmBannerAutoResetAckAlarms=0
ServerName=127.0.0.1	AlarmBannerKeepSortingBySeverity=0
[ServerMisc]	**[Enterprise]**
EmailServer=SMTP_Server	EnableProducerRedundancy=0
EmailFrom=sender@domain.net	SendUpdatesInBackupMode=1
EmailSubject=Alarm	
EmailUser=recipient@domain.net	
EmailPassword=	

EmailOnAck=0 EmailOnNew=0 EmailOnClearAck=0 HBInterval=5000 HBTimeOut=10000 SSPath=C:\Program Files\ClearControls\ClearView EnableSS=0 SSFile=	

Note: The ClearView Server should be shutdown and restarted after changing any project settings in the *.cvp file.

Project Files

File Types

The following table lists file types of files used by ClearView

File Type	Description
*.cvp	File contains project settings. A corresponding *.ccp file must be located in the same directory. Open the *.cvp file for normal operations
*.ccp	A binary graphics file. Open this file type in cases where the *.cvp file does not exist or is corrupt. To select this file type, change the Save As Type drop-down list in the Save As dialog box. A corresponding *.cvp file will be created automatically when the project is saved.
*.arg.xml *.arg.stg	Files containing GIS configuration. Warning: do not modify those files manually.
*.zip	The zip file contains at least the *.cvp and *.ccp files, and may contain more files. The ZIP file is automatically created when a user saves a project. To select this file type change the Save As type drop-down list to “All Files” in the Save As dialog box

Creating a Project File

1. On the **File** menu, select **New.** – ClearView Project Settings window will appear.
2. On the **Project** tab, click the button shown under Project File Name.

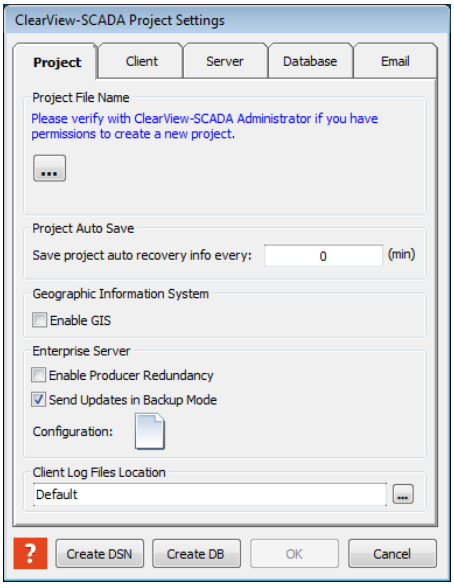


Figure #2.1

3. Use the Windows Save As dialog box to browse for a location.
4. Enter a file name.
5. Press **Save.**
6. Enter project auto save frequency (in minutes).

Note: The backup files are saved in project root directory.
Folder Name: Project Name_Screen (backup).
File(s) Name: Project Name_mm_dd_yy_hh_mm_ss

7. Check Enable GIS if necessary, see SECTION 13 for more details on ClearView Geographical Information System.
8. Check Enable Producer Redundancy if necessary, see SECTION 19 and SECTION 20 for more information on ClearView Redundancy and ClearView Enterprise Server
9. Enable or disable sending updates in backup mode by checking/unchecking Send Updates in Backup Mode
10. Configuration - ClearView Enterprise Server configuration (file)

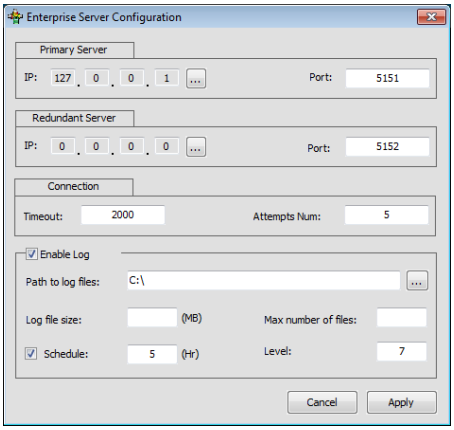


Figure #2.1A

11. By default ClearView Client's logs are stored in the same directory where ClearView was installed. To override default location change the **ClearView-SCADA Client Log files location** path

Note: After you press OK the project file is not saved yet. It will only be saved when the project set up is complete and the user selects Save from the File Menu. See Menus. It is possible to open a file with the extensions *.cvp, *.zip or *.ccs.

Client Settings

The **Client** tab on the Project Settings dialog box defines how ClearView Client stations may log in or out and how often the screens are updated.

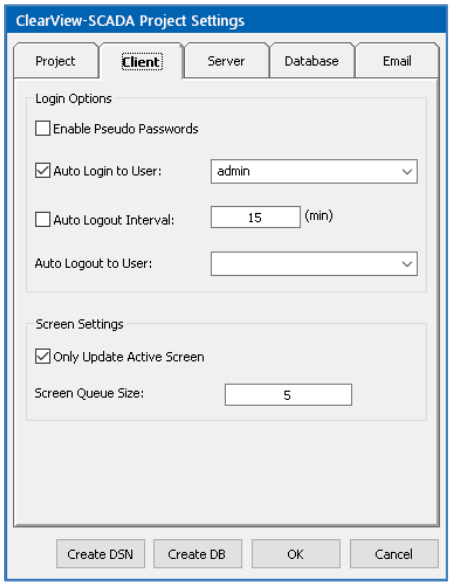


Figure #2.2

Login Options

Enable Pseudo Passwords	Check this option to make the Login dialog box to only require a valid password to log into the system.
Auto Login to User	Check this option to auto3matically log in the system using the selected user account.
Auto Logout Interval	Check this option to automatically log out a user after an idle interval of time has expired.
Auto Logout to User	Select this option from a drop-down list to allow a particular user account to be logged in after an automatic logout has occurred. Tip: You can create a user account called No User with no security privileges for use with the Auto Logout to User account.

Screen Settings

Only Update Active Screen	Several screens may be opened at once; however, only the top screen is visible. To save system resources, only the top-most visible screen will be updated with the latest tag values and animation.
Screen Queue Size	Microsoft's Operating Systems have limits to the number of objects opened in its window environment. For this reason, it may be necessary to limit the number of ClearView screens to load/open in a project. The **Screen Queue Size** determines the maximum number of screens allowed to be loaded into memory. If this value is set to 0, then all projects screens can be loaded into memory.

Server Settings

The **Server** tab on the Project Settings dialog box defines what ClearView Server to connect to and what, if any, server script files to run on the Server.

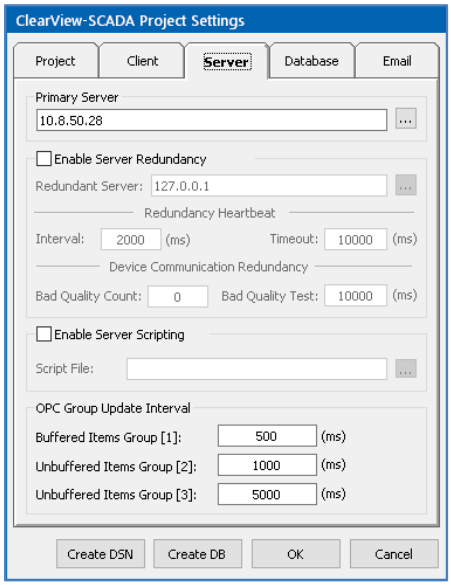


Figure #2.3

Primary Server

Specify the ClearView server name to which you want to connect. Click the button to the right to browse for the running servers.

WARNING: Do not enter or select a server if the ClearView database has not yet been created or setup.

Note: The server name can also refer directly to the IP address (XXX.XXX.XXX.XXX) of the machine where the ClearView Server is running.

Redundancy

Specifies ClearView Server redundancy, see SECTION 19 for more information on ClearView Redundancy.

Server Scripting

Use the Server tab to select the Enable Server Scripting check box. Select a CCS file in the Server Script File setting. A CCS file is created using the Server Script Editor. This editor is only accessible in the Application Explorer of ClearView after a project has been created.

If you have created and saved a server script file, click the right button to browse and select the script file (*.ccs) that will run after starting the ClearView Server.

OPC Update Interval

The OPC Update Interval controls how fast, in milliseconds, your OPC Server will notify tag value changes to the ClearView Server and ClearView Clients.

Buffered Items Group-1 reports all data on change (Immediate I/O). It is used for fast changes up to 4 milliseconds.

Unbuffered Items Group-2 reports last known value. It is used for Controls like preset values, Breakers, etc.

Unbuffered Items Group-3 reports last known value. It is normally used for meter values such as Voltage, Power, Power Factor, etc.

A Data Point can be assigned to a particular group when the corresponding Tag is created in ClearView.

Create Database

Note that Create Database Dialog below is designed to work with MS SQL and MS Access Database. Oracle Data base can be created by using sql script provided with ClearView, see Create Oracle Database below.

ODBC is a type of driver that provides access to a variety of different databases. Most MS Access and SQL SERVER databases are accessed via an ODBC driver. If you have an Oracle database installation, you may use an Oracle driver. The data source is an instance of an ODBC connection. You can define data sources via Control Panel > Administrative Tools >Data Sources (ODBC). The data sources that are currently defined will be displayed in the Data Source drop-down list. During a ClearView installation a ClearControls data source is created.

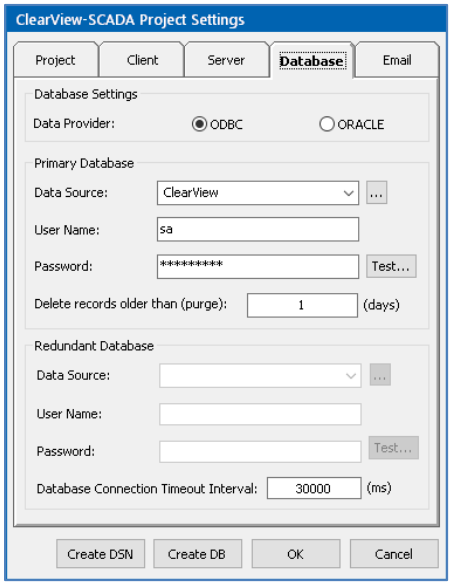


Figure #2.4

User Name and **Password** are not required for MS Access. These settings are required for SQL SERVER and Oracle. Click **Test…** to see if the database connection has been established with your configured settings.

Create DB will display the Create Database dialog box shown on Figure #2.5 below.

Source/Destination

The source file will have the (*.zip) extension. This file has to be created by the ClearView backup utility. The backup utility is accessible via the File menu (File > Backup).

The destination folder can be any folder.

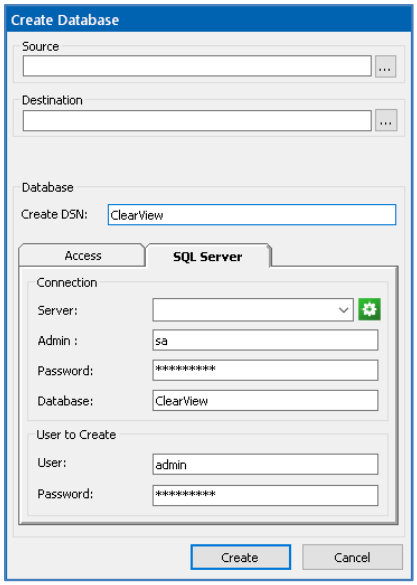


Figure #2.5

Restore Database

If the backup database was selected during the backup process, then the zip file (selected for the Source setting above) contains a database called structure.mdb. Check **Restore Database** to restore the data to a new database specified in the following settings.

Create DSN

Creates a new Datasource Name. A DSN is an instance of an ODBC driver, for example, an MS Access driver that points to a MS Access database file (*.mdb). This new DSN will be required in the Database Settings section for the project setting named Datasource. The Datasource setting will be saved in the ClearView project file (*.cvp).

SQL Server Database

These settings are for the new database that will be created. There are two types of databases that can be created: MS Access and SQL Server.

Server	The name of the computer running SQL Server.
Admin/Password	A user name with administrative rights on the SQL Server
Database	The name of the database to create on the SQL Server. A user will be created for this database. For general connection non-administrative duties, it is recommended this user be specified when connecting to SQL Server. Specify this user in the ClearView username/password project settings in the section Database Settings.
User to Create	Creates another data base user.

Access Database

If creating a MS Access database, specify a MDB filename. MS Access databases do not require a username/password to log in.

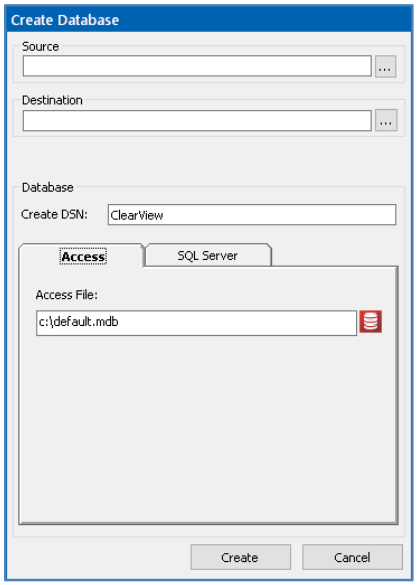


Figure #2.6

Creating Oracle Database

Before configuring database parameters user must create a default ClearView database (an Oracle user). ClearView installation provides *CVDBOracleScript.sql* script file that is located in \\ClearControls\ClearView\bin directory. To run the script user must have sufficient Oracle administrative privileges to be able to create a new database.

By default the script will create the Database named *clearviewora* with password *clearview.* To create a database with a different name please modify script as needed.

Note: The script is designed to run only once on the same database. In case of script errors and if the database was already created the database has to be dropped before running the script again.

Create New DSN

Click on Create DSN button

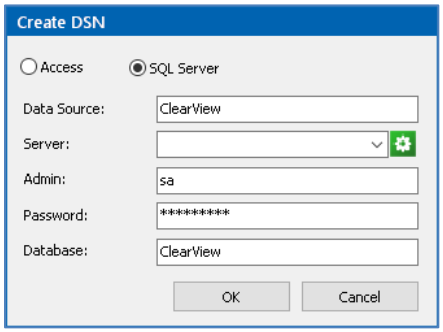


Figure #2.6A

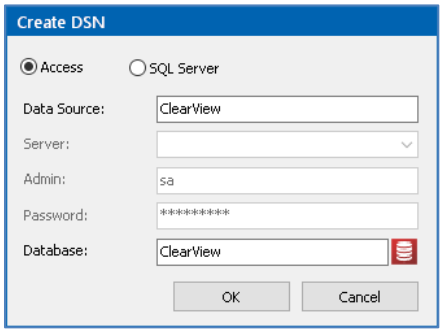


Figure #2.6B

Email Notification Setup

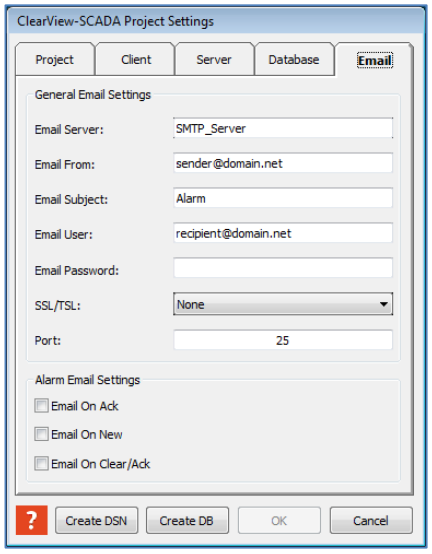


Figure #2.7

Email Server	Provide an SMTP mail server account.
Email From	Enter a return email account.
Email Subject	Enter the subject for this email.
Email User	Enter a valid email account on the specified SMTP server.
Email Password	Enter the corresponding password to this email account.
SSL/TLS	Enable/Disable SSL/TLS for encrypted connection. Note: For Port 465 use Direct SSL/TLS for anything else use START TLS (default)
Port	Outgoing Server SMTP Port number
Email On	Three alarm events that trigger an email are Acknowledge an alarm (Ack), New Alarm (New), and Alarm cleared and acknowledged (Clear/Ack).

More than one database per a setup

Typically ClearView Client and ClearView Server are configured to read Tags and Alarms form the same data base, meaning that they both work with the same sets of Tags and Alarms. ClearView-SCADA provides also ability for Client and Server to read Alarms and Tags from different data bases.

A typical use case for that is as follows.

- There is more than one Client in ClearView-SCADA setup
- Some of Clients need only a subset of configured Tags and Alarms

To avoid unnecessary network traffic and minimize Client's CPU consumption it is recommended to create the configuration where ClearView server is working with a database containing all Tags and Alarms, and Clients that need only small subsets of Tags and Alarms are reading them from other databases.

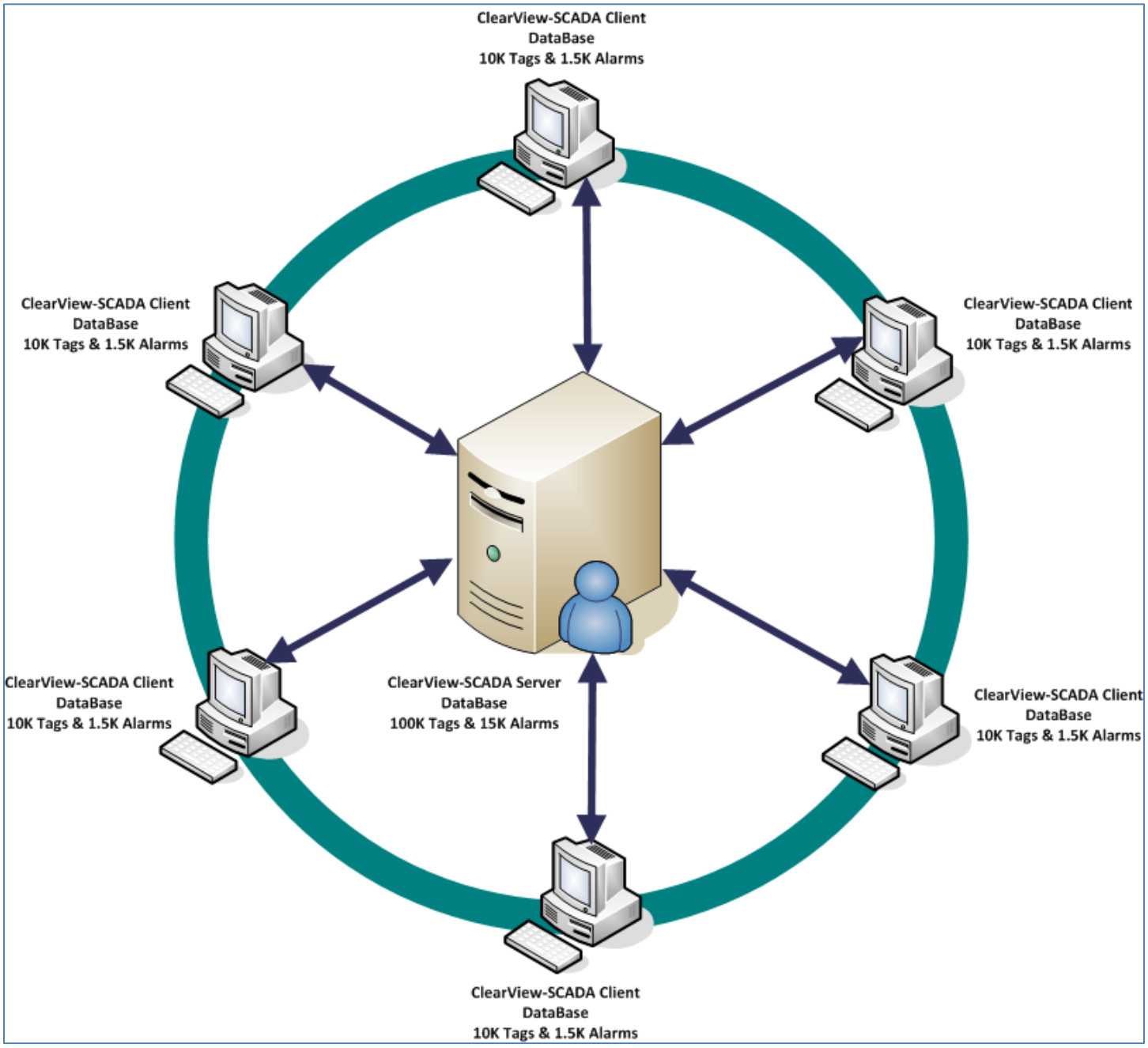


Figure #2.8

A configuration like this can be achieved by creating different projects for Clients and Server and pointing them to different data bases. The Tags or Alarms in Clients' databases must exist in Server's data base, otherwise the missing tags would receive a bad quality.

SECTION 3

Tags

A tag is a name used to refer to an associated data point, the value of which can be displayed directly or indirectly, stored for retrieval at a later time, or read from or written to an IO device such as a PLC. The value of a tag can be based on an address in a PLC or derived from a function that may involve several defined tags.

ClearView Server communicates to field devices using OPC 2.0 protocols. There is an intermediate layer between ClearView Server and the PLC device known as an OPC Server. ClearView Server writes a tag value to an OPC Server and the OPC Server writes this value to a PLC device.

ClearView provides an OPC server called CVOPC for specific PLC devices. There are other OPC Server options, for example Kepware, which provide communication to a myriad of PLC devices.

Note: An OPC Server maintains its own set of tags. ClearView tags can be exported to a comma-separated text file to facilitate the creation of tags for an OPC Server.

Tag Database

A summary of tag names properties in the ClearView database can be viewed using the Tag Database window. Once open, use the scroll bars or arrow keys to select or view the tags listed in the database, or enter a filter string in the Filter Entry field. Access to the Tag Database is controlled by the security setup.

To view the Tag Database:

1. Open **Application Explorer**
2. Expand the **Setup** folder
3. Double-click **Tag Database**

Note: The Tag Database can also be viewed from the **Project** menu, pointing to Setup, and clicking **Tag Database**.

Tag Database Menu

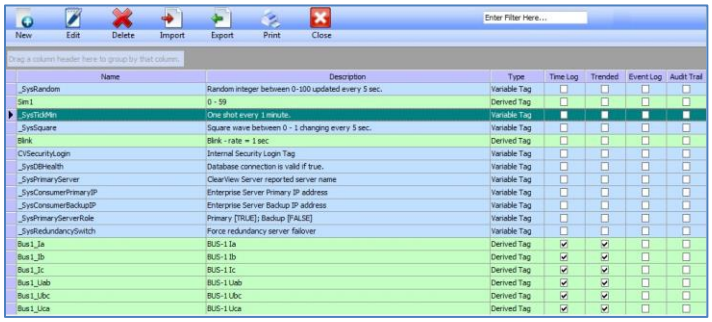


Figure #3.1

New	Opens the Tag Builder for adding new tags.
Edit	Opens the Tag Builder for editing the selected tag.
Delete	Deletes the selected tag from the database
Import	Imports a *.csv file into the Tag Database. All fields must match when manually editing a *.csv file. Imported tag names that match the existing tag name will have its fields updated, while new tags will be appended to the database.
Export	Exports the entire tag database to a *.csv file. An Export dialog box will appear. If you have special characters such as “#” then select that option and specify your own delimiter,
Print	Opens the Windows Print dialog box to print the Tag Database. The printout will show similar columns as shown in the Tag Database window.
Close	Closes the Tag Database window.

Tag Types

ClearView supports three types of tags:

- **OPC Tags**. ClearView Server communicates to IO through common OPC servers. The actual tag value may represent any data type: string, real, long, short, floats, Boolean, or other as defined in the OPC server.
- **Derived Tags.** Derived tag values are the result of an expression or formula consisting of other tags in the database and/or combined with available VB Script functions. The expression is built with the Expression Builder. The tag value may be displayed, trended, logged, used for triggering alarms, and/or written to an IO device if assigned an OPC address.
- **Variable Tags.** Variable tags are tags not tied directly to an OPC Server. They represent constants or system information that is passed between server and client PCs.

To create a Tag, press **New** from the Tag Database window. The Tag Properties dialog box appears.

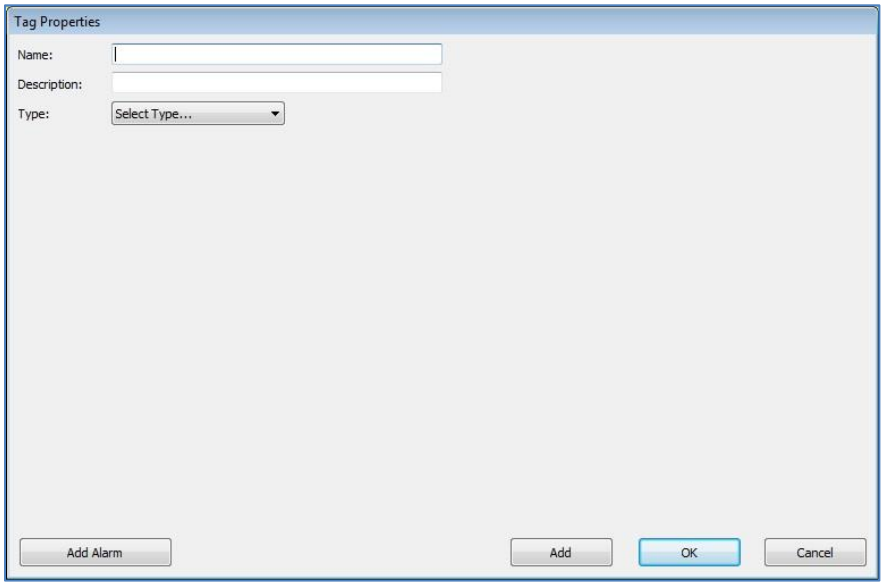


Figure #3.2

1. Enter a **Tag name** in the Name field. A Tag name can be 256 alphanumeric characters long. No spaces or dashes are allowed in the tag name. The name should only use alphanumeric charters and the underscore character.
2. In the **Description** field, enter a description representing the IO point. Field length is 256 characters.

WARNING: Do not use the comma character in the description or expression fields. If you do, the importing/exporting to a comma separated variable file (*.csv) will not function correctly.

Creating an OPC Tag

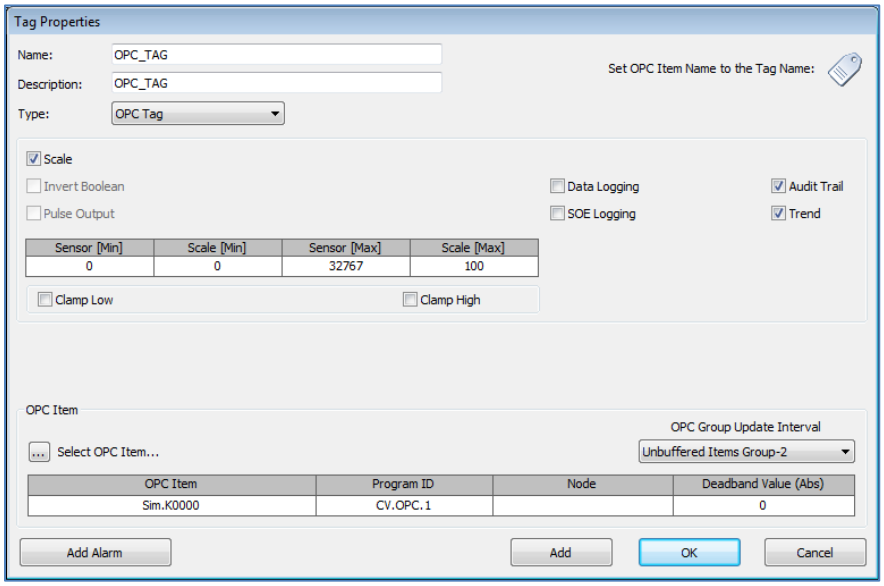


Figure #3.3

When you select **OPC Tag** in the Type field, the following options will appear:

- To scale the OPC tag in ClearView, click **Scale** and fill in the **Raw Min**, **Raw Max**, **Scale Min**, and **Scale Max** fields.
- To clamp the values within the scaled ranges, click **Clamp Low** and/or **Clamp High**.

- To log the tag value to the History table at minimum interval, click **Log** and enter a log **Interval**. The default interval value is set at 60 seconds.
- If you want to log a tag only during the time that a condition or event is active, then click **Event Logging** and select a **Tag** that evaluates to a Boolean type.
- If you want to log sequence of events data (SOE logging) click SOE Logging checkbox.

  Note: Logged tags are written to the database if the value changes during the log interval. If the value doesn't change for over a day, then there will be no additional repeated values recorded during that day. This also applies to event correlated logged tags

- To allow a tag to be updated on a real-time trend, click **Trend**.
- To enter the OPC reference fields, press the **Select OPC Item** to open the OPC Browser.
- To apply a deadband to the OPC tag value, enter an absolute value in the **Deadband Abs Value** field.
- To save your changes, press **Add** to save the tag and move to a new tag, or press **OK** close.
- Add Alarm button opens Alarm Properties interface and allows specifying alarm conditions for the tag. The tag will be saved to the Tag Database at this point.

Tag Property: OPC Pulse Output

The tag Pulse Output feature applies only to ClearView-SCADA Boolean OPC Tags. To define Pulse Output the Pulse Output check box must be checked in Tag Properties, see Figure #3.4 below.

Short Pulse Operation:

- The initial state of the Pulse Output is False (0)
- User writes True (1) to Pulse Output
- ClearView-SCADA Server confirms that value of TRUE (1) was written to OPC Server
- ClearView-SCADA Server writes value of FALSE (0) to OPC Server

Note: The timing of change depends on device respond time, OPC Server poll time and network performance.

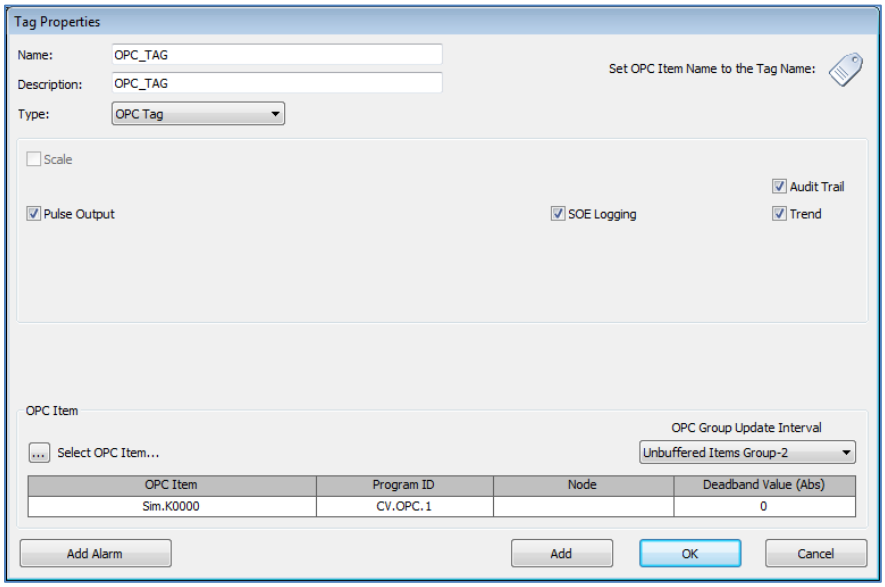


Figure #3.4

Creating a Derived Tag

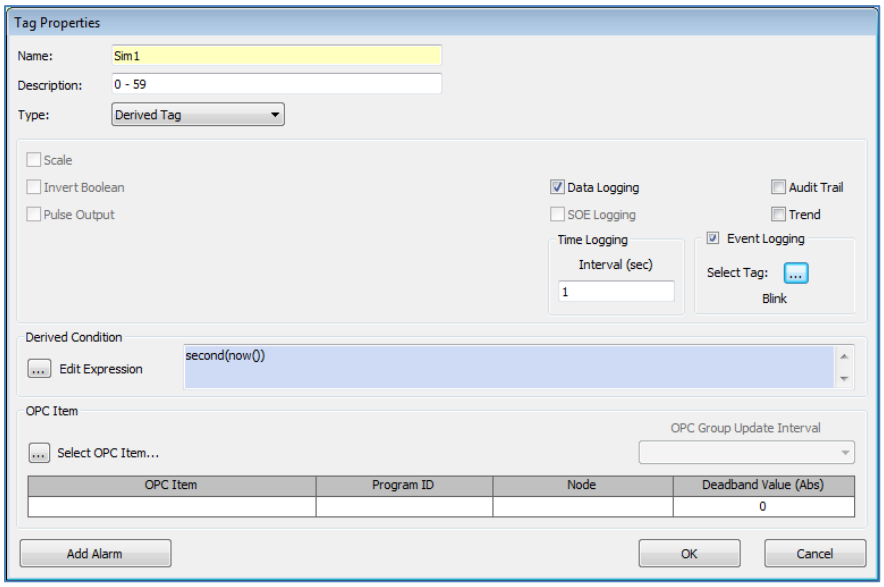


Figure #3.5

1. On the Tag Builder dialog box, select **Derived Tag** in the Type drop-down list.
2. To save your changes and move to a new tag, press **Add**, or press **OK** to save and close.

Creating a Variable Tag

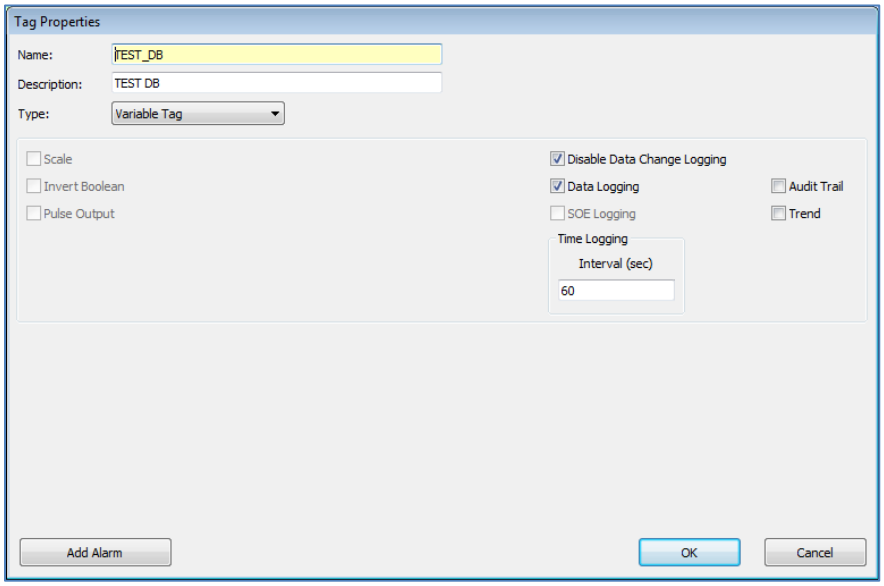


Figure #3.6

The variable tag type is a system memory tag. It is not tied to any existing tag or OPC IO. Its value can represent any data type. The length of the value field is limited to 256 characters.

1. On the Tag Builder form, select **Variable Tag** in the Type dropdown list.
2. To save your changes and move to a new tag, press **Add**, or press **OK** to save and close.

Trending a Tag

Check the trending option in the tag builder to allow the tag to be listed on the Trend Viewer form. The Trend Viewer form is accessible via the Application Explorer in the Trends folder. A trend is a graphical view of tag values over a specific period of time. Real-time trends are updated every 500 milliseconds. Historical trends must access the History table of the ClearControls database to retrieve historical data values for a tag.

To turn on trending of a tag:

1. Create new OPC or derived tag.
2. Select the **Trend** property.
3. After tag set up, restart ClearView Server.
4. Open the **Trend Viewer** that is available via the Application Explorer in the Trends folder.

Trend Viewer

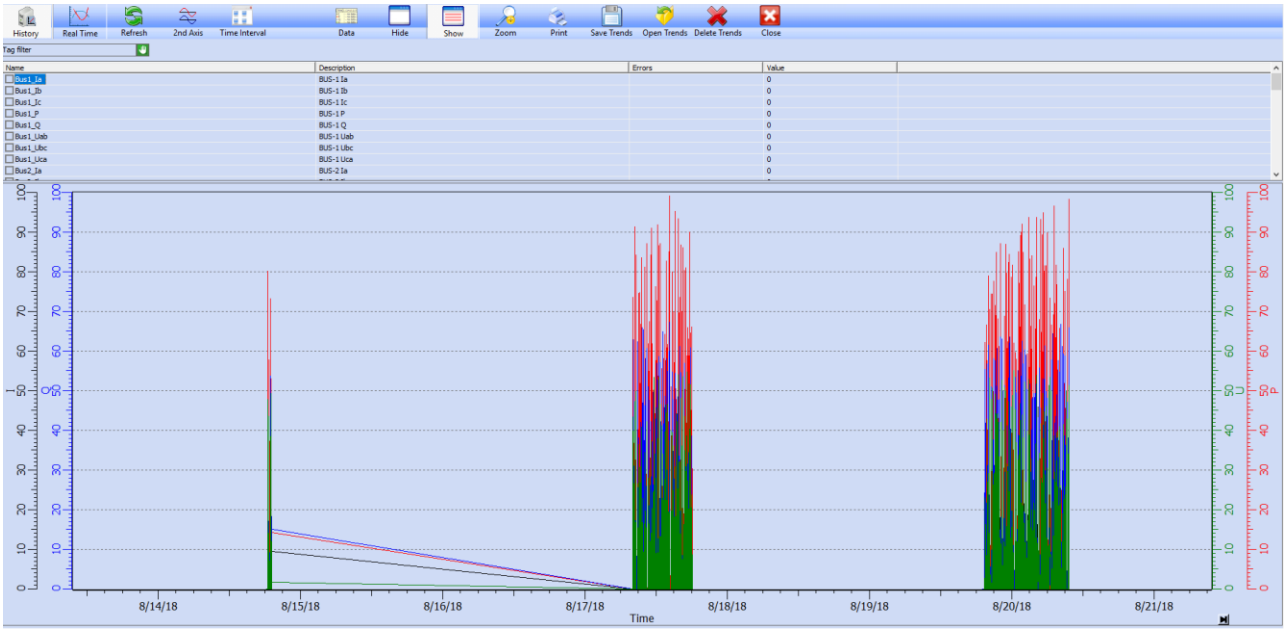
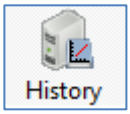
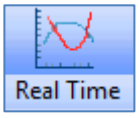
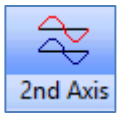
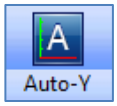
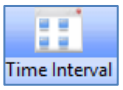
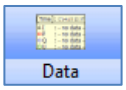
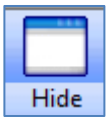
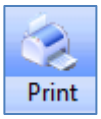
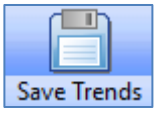
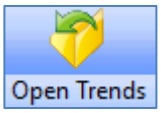
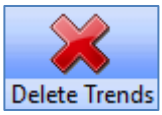
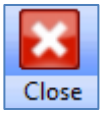


Figure #3.6A

	Opens and views historical data
	Opens and views Real-Time data

	Adds 2nd X Axis
	Auto Resize to max “Y” value
	Set’s the trend time interval for selected records (see Figure 3.6B)
	Enables curve data window view
	Hides the tag section interface
	Makes tag section interface visible
	Enables trend zoom functionality
	Prints the trend data
	Saves trend template (see Figure 3.6C)
	Opens trend template (see Figure 3.6D)

	Deletes trend data
	Closes Trend Viewer interface

Time Interval

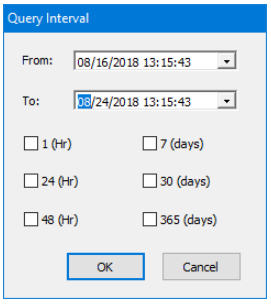


Figure #3.6B

Save Trend

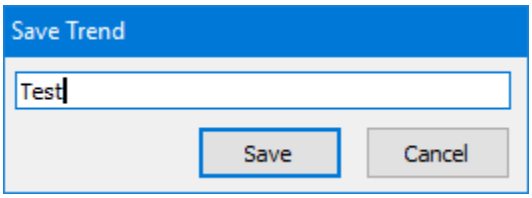


Figure #3.6c

Open Trend

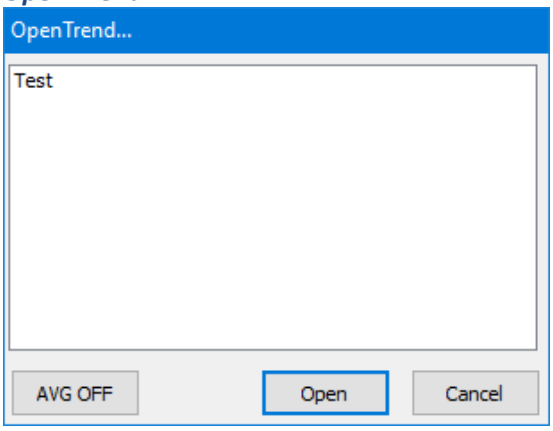


Figure #3.6c

Note: “AVG ON/OFF” enables or disables pen averaging

Data Logging

It is possible to log tag values based on time, interval, or event. The logged values are visible via log reports. The Data Log Report is accessible from the **Application Explorer** in the Reports folder. While setting up a tag, specify in seconds how often you want to log tag values.

Log Type	Description	Applicable Tag Types
Data Log	Time based logging (time entered in seconds)	OPC Tags, Derived Tags and Variable Tags
Event Logging	Time & Event based logging (Trigger tag must be specified)	OPC Tags, Derived Tags and Variable Tags
SOE Logging	Sequence of Events logging (logs array of data not based on time	OPC Tags ONLY
Disable Data Change Logging	Logs data regarless of tag change	Variable Tags ONLY

Note: It is possible to overburden a system by setting the log interval too short. The log interval should be based on computer processing power and the total number of tags logged. Event logging allows a tag to be logged whenever an event is occurring. The event is actually another tag. Whenever the event tag resolves to true, logging will occur.

Note: A tag resolves to true for any value except for zero. An event logging causes a tag to be logged once when the event has occurred.

Importing/Exporting Tags

Exporting tags is a convenient way to move tags from one database to another in CSV format. You may also modify a large number of tags at once via Microsoft Excel and then re-import them into ClearView. Finally, exporting tags to a CSV format can aid in the creation of OPC Server tags.

To export tags:

1. Navigate to the Tag Database, which is accessible via the Application Explorer in the Setup folder.
2. Click **Export** on the Tag Database toolbar. An Export dialog box will appear.
3. Accept the default options in the Export dialog box and click **Export**.

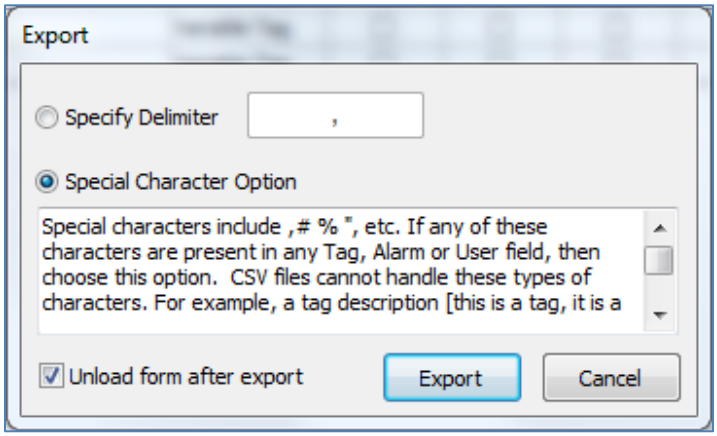


Figure #3.7

4. Specify a file name in the **Save As** dialog box.

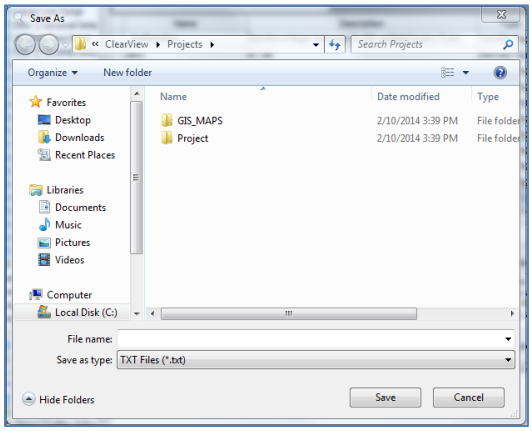


Figure #3.8

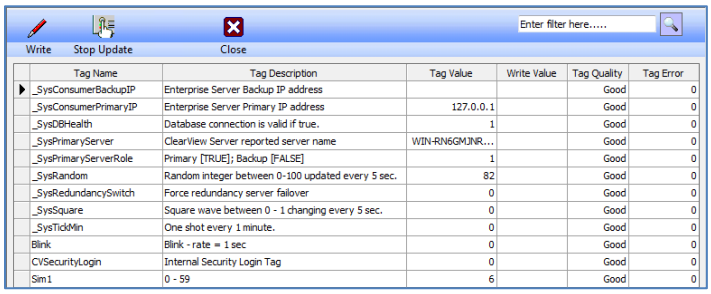
Importing tags requires a strict format. Any manual modifications to a CSV file may cause an import to fail. Therefore, make sure all fields are correct before importing a CSV file.

1. Click **Import** on the Tag Database toolbar.
2. Select a CSV file in the Open dialog box.

After importing, ClearView will either report an error or the number of tags that were updated and/or inserted.

Data Spy

The DataSpy allows a user to read and write values of any tag. The Quality and Error of a tag are also visible. If the quality is bad for a tag that is of type OPC then corrective action must be taken. This may involve checking the memory location on the PLC device or the OPC Server configuration.



Tag Name	Tag Description	Tag Value	Write Value	Tag Quality	Tag Error
_SysConsumerBackupIP	Enterprise Server Backup IP address			Good	0
_SysConsumerPrimaryIP	Enterprise Server Primary IP address	127.0.0.1		Good	0
_SysDBHealth	Database connection is valid if true.	1		Good	0
_SysPrimaryServer	ClearView Server reported server name	WIN-RN6GMJNR...		Good	0
_SysPrimaryServerRole	Primary [TRUE]; Backup [FALSE]	1		Good	0
_SysRandom	Random integer between 0-100 updated every 5 sec.	82		Good	0
_SysRedundancySwitch	Force redundancy server failover	0		Good	0
_SysSquare	Square wave between 0 - 1 changing every 5 sec.	0		Good	0
_SysTickMin	One shot every 1 minute.	0		Good	0
Blink	Blink - rate = 1 sec	0		Good	0
CVSecurityLogin	Internal Security Login Tag	0		Good	0
Sim1	0 - 59	6		Good	0

Figure #3.9

Note: Tag Quality and Tag Error fields are relevant to OPC tags only.

Writing to a Tag with DataSpy

To write a tag value:

1. Locate the tag in the grid.
2. Click on the Write Value field of the tag.
3. In the Write Value dialog box, Figure #3.9, enter a value and click **OK**.
4. Click **Write** on the DataSpy toolbar.

The new value will appear in the Value box for this tag.

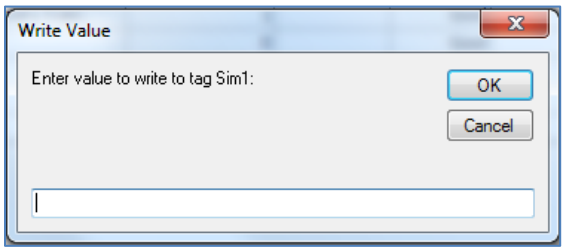


Figure #3.10

QuickView

QuickView is a smaller version of the DataSpy. It is useful for viewing and comparing a small number of tags.

To view a tag:

1. Click the **Add Tag** button on the QuickView toolbar.
2. The **Select Tag** dialog box will appear.
3. Select a tag from the list and click **Select**.

The tag will now be listed in the QuickView window. You may select another tag using the same procedure.

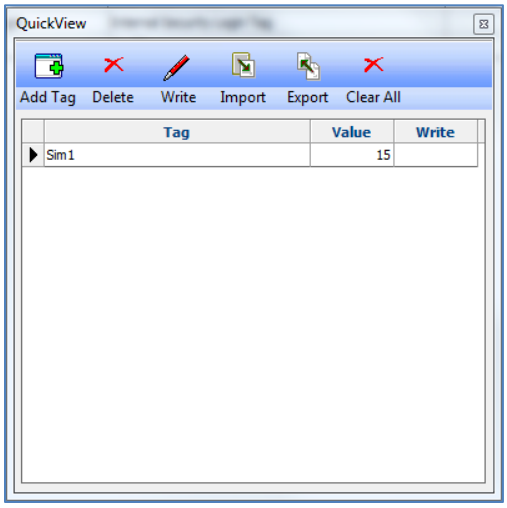


Figure #3.11

- To delete a tag from QuickView, click **Delete** on the toolbar.

Import/Export QuickView Tags

Often the same group of tags needs to be listed in the QuickView window. Instead of selecting each tag individually, it is possible to save the current tag list configuration by clicking **Export** on the QuickView toolbar. The tag list is saved to a CSV file. Next time you start ClearView, you can then simply import the CSV file

Putting a tag offline/online

Starting from ClearView version 10 the user is capable of putting any OPC Item offline (Local Override).
In order to put the OPC Item (tag) offline the string "**IO_OFFLINE**" should be written to selected/required tag
The quality of this tag would change immediately to **216** Good [Local Override]

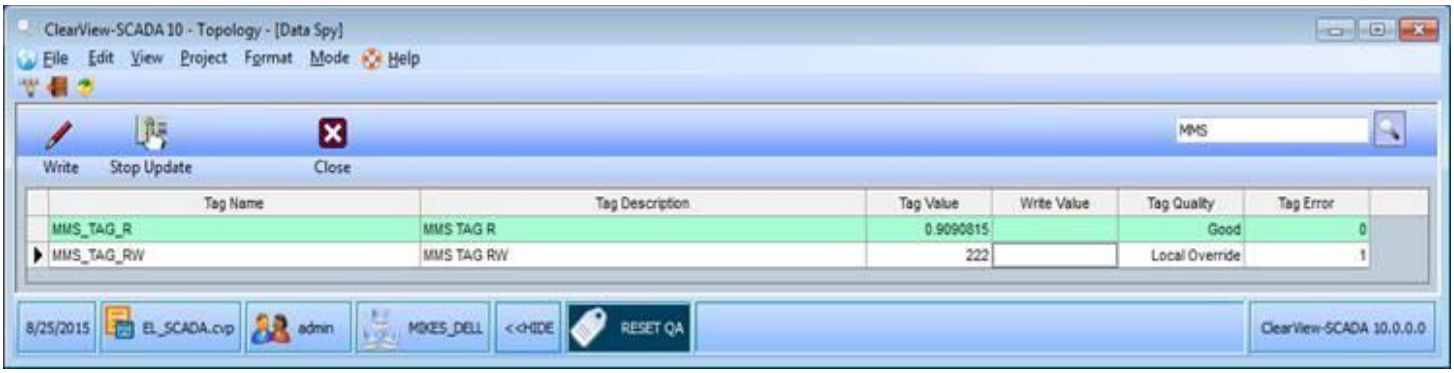


Figure #3.12

Now any value can be written to this tag but it would be NO updates coming to OPC Server.

On ClearView client DataSpy the color of this tag would change (white background & black text) and column "Tag Quality" would receive a text "Local Override".

If the user needs to put the tag back to normal the string "**IO_ONLINE**" should be written to the value of this tag.

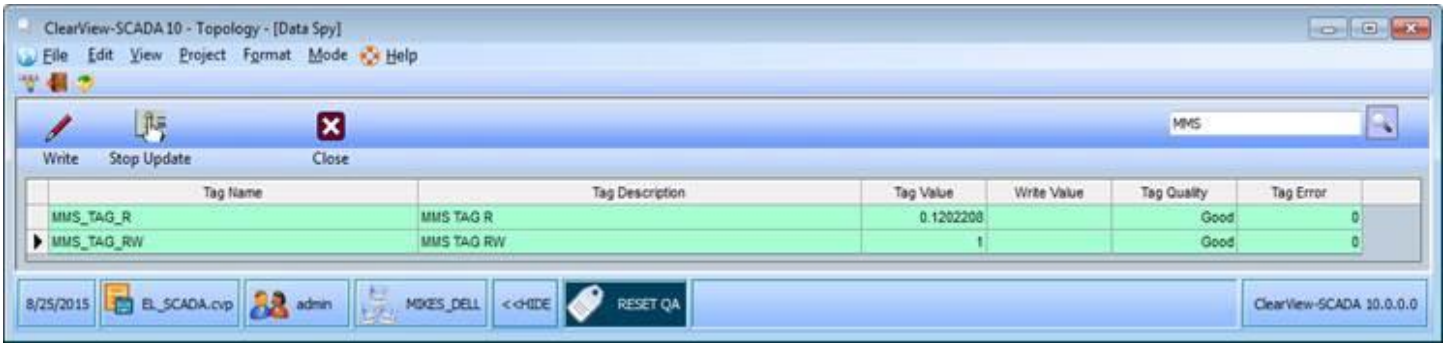


Figure #3.13

The quality would immediately change to **192** [Good] and the value would be updated from OPC Server.

Note: The strings "**IO_OFFLINE**" or "**IO_ONLINE**" are not case sensitive.

SQL Builder

The purpose of the SQL Builder is to perform queries and operations on a database. The grey textbox in the center of SQL Builder dialog box will contain the SQL statement to execute. It is not possible to manually type in the statement. It must be built using the SQL Expression Builder. Users should have a basic knowledge of Structured Query Language before using SQL Builder.

- To view the SQL Builder, open the Application Explorer and select **SQL Builder** in the Setup folder.

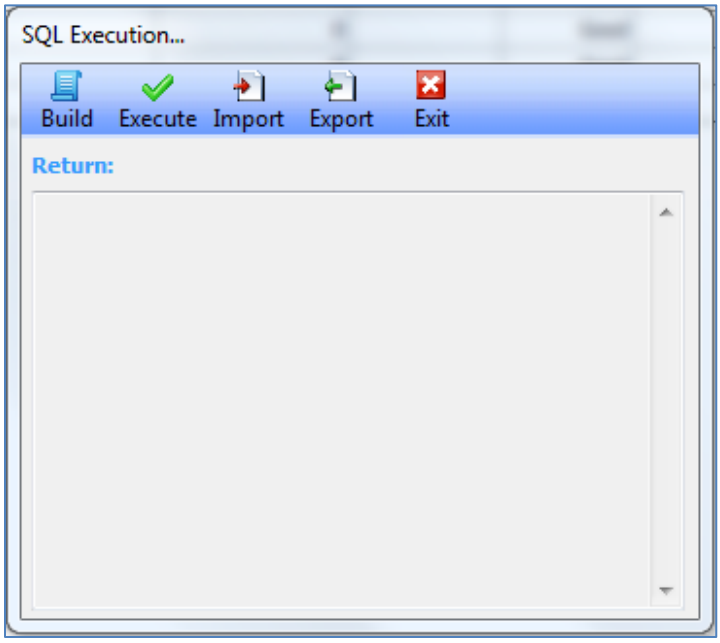


Figure #3.14

To build the SQL statement, perform the following steps:

1. Select database.
2. Select a command: Select, Insert, Update or Delete.

WARNING: Please be aware that update and delete commands will update or delete the record permanently.

3. Select a table.
4. Enter a condition clause.

To execute the SQL statement:

- Alter or delete database records via the SQL Builder

To import the SQL statement:

- Import a saved SQL statement from a SQL (*.sql) file.

To export the SQL statement:

- Save current statement to an SQL file

SECTION 4

Alarms/Events

An alarm is typically a notification of an abnormal condition that exists in your process. An event is a notification of a change in a system condition, for example a user logging on or a ClearView Client connecting to the server. ClearView Server distributes Alarms and Events between clients and server through an alarm object that resides on both clients and server machines. The alarm object contains the collection of alarms created by a user and the events monitored by the system. As alarm conditions change or system events occur the server notifies all connected clients of the change.

The status of alarms can also be distributed remotely to offsite personnel. ClearView can email through an SMTP email server to maintenance or other personnel responsible for operating the system. Alarms and events are logged through the database server to a common table called Event_Log. Historical or current Alarms and Events can be viewed on a ClearView Client using the Alarm/Event Viewer. Current alarms can also be viewed and acknowledged using ClearView Alarm Banner ActiveX control.

To distinguish between active alarms and the names of individual records contained in the alarm collection, you can use the Alarm Tag name to identify the configured record.

An Alarm Tag is added to the database by entering its properties though the ClearView Alarm Properties dialog box. The alarm configuration includes setting the Alarm Tag name, description, alarm group, the conditional setting that raises the alarm. It may also include who should be notified if the alarm condition changes. Alarm Tag records are stored in the database server's Alarms table.

Each ClearView Client station must have its Alarm Groups registered on the PC to receive notification of the alarms assigned to the group. This allows selected groups of alarms to be viewed by selected Client stations.

When an alarm occurs it is also possible to create an entry in the Log Book.

Alarm Database

A summary of alarm tags in the ClearView Server database can be viewed using the Alarm Database window. The Alarm Database window is used to add, edit, delete, import/export, and print the alarm tags contained in the Alarms table. Once open, use the scroll bars or arrow keys to select or view the alarms listed in the database, or enter a filter string in the Filter Entry field. Access to the Alarm Database is controlled by the security setup.

Note: Alarm Tag names are ordered alphabetically, **not** numerically.

Viewing the Alarm Database

1. Open Application Explorer.
2. Expand the Setup folder.
3. Double-click **Alarm Database**.

Tip: The Alarm Tag Editor can also be accessed from the Project menu: Select Setup and click Alarm Database.

Understanding the Alarm Database Menu

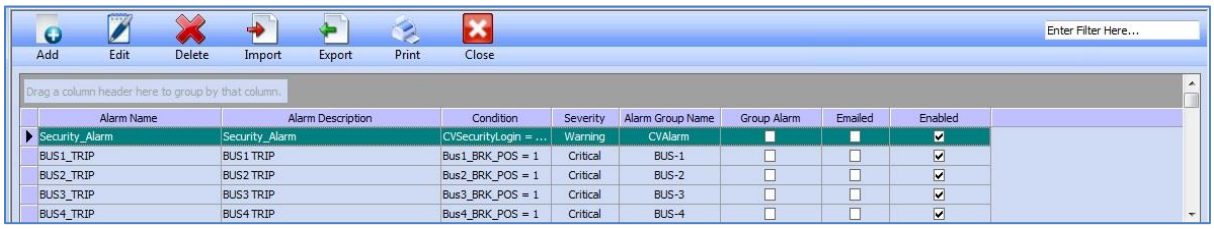


Figure #4.1

Add	Opens the Alarm Properties dialog for adding new alarms.
Edit	Opens the Alarm Properties dialog for editing the selected alarm.
Delete	Deletes the selected alarm record from the database.
Import	Imports a *.csv file into the Alarm Database. All fields must match when manually editing a *.csv file. Imported alarm names that match an existing alarm name will have its fields updated, while new alarm names will be appended to the database.
Export	Exports the entire alarm database to a *.csv file. An Export dialog box will appear. If you have special characters as a separator such as “#”, then select that character as an option.
Print	Opens the Windows Print Dialog box to print the Alarm Database. The printout will show similar columns as shown in the Alarm Database window.
Close	Closes the Alarm Database window.

Working with Alarms

The Alarm Name and the conditions that define the alarm are defined in the Alarm Properties window.

To add an alarm or edit alarm properties:

1. Open the Alarm Database.
2. Select an existing alarm and click **Edit** or click **Add** to create a new alarm.

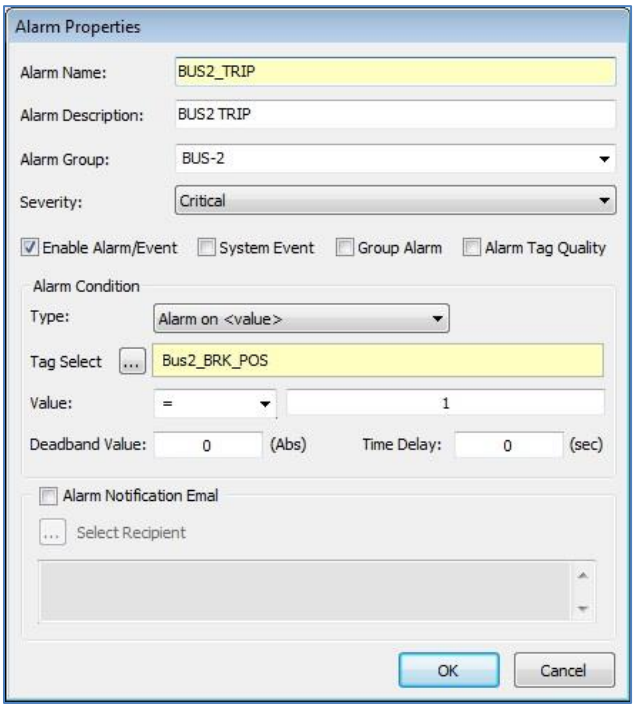


Figure #4.2

- **Alarm Name**. A text string, maximum 256 characters, consisting of alphanumeric characters. No spaces or hyphens or any non-alphanumeric character is allowed except underscore.

  Tip: ISA standards provide tag-naming conventions similar to the device/tag that raises the alarm, using a combination of prefixes and suffixes to define the particular condition that activates the alarm.

  Example: LI607 may be the analog level indicator for device 607. Therefore, LAHH607 would be the alarm tag name for the High-High Level Alarm or device 607.

- **Alarm Description**. A text string, maximum 256 characters. The alarm tag name and its description are communicated when an alarm condition occurs

  Note: Do not use commas in the description field, as this will cause errors when importing/exporting the alarm table

- **Alarm Group**. Select from existing Alarm Group names in the dropdown list or enter a new alarm group name.

  Note: Alarm notifications will not be received on a ClearView Client PC until the associated Alarm Group is enabled/registered. See Enabling Alarm Groups.

- **Alarm Tag Quality**. Specifies if alarm would be generated based on Tag Quality rather than on Tag Value.

  Note: Good Tag Quality = 192

- **Severity**. Classify the alarm by selecting one of the available severity types: Warning, Critical, Shutdown, or Interlock plus 32 types named Priority-1, Priority-2, … , Priority-32.
- **Enable Alarm**. Check this to enable/disable alarm notification for this alarm.
- **System Event.** Check this to enable/disable system events notification. When selected the new System Group will be created with [SE] appended to the group name entered.
- **Group Alarm.** Creates a group alarm. A Group Alarm is an alarm that can be queried and acknowledged as an entire group.
- **Alarm Condition.** An alarm becomes active when the Selected Tag value meets the predefined condition. Likewise, an alarm becomes inactive when the Selected Tag value does not meet the defined condition. For discrete or Boolean (0=false=OFF, 1=true=ON) type Tags, select **Alarm When Tag is OFF** or **Alarm When Tag is ON**.

  For analog type Tags, select **Alarm On Value**, then chose a **Conditional Operator** and enter the **Limit** value when the alarm is to be active. To reduce alarm nuisances, enter a **Deadband Absolute Value**. (A value of 0 deadband applies no deadband to the alarm evaluation.)

- **Alarm Notification**. If the Alarm needs to be emailed to users, enable the email notification and select the users (Recipients) to be notified. For more information, see Emailing Alarms.
- **Conditional Operators.** You can use the following conditional operators:

Operator	Description
=	Equal to
>	Greater than
<	Less than
=>	Greater than or equal to
<=	Less than or equal to
<>	Not equal to
In Range	The value is in a specified range
Not In Range	The value is not in a specified range

- **Alarm Deadband Examples**

Deadband	Condition	Alarm On	Alarm Off	Alarm Equation
10	> 100	> 110	< 100	> 100 + D
10	< 100	< 90	> 100	< 100 – D
10	= 100	< 100 + 5 or > 100 – 5	<> 100	< 100 + D/2 or > 100 – D/2
10	<> 100	> 100 + 5 or < 100 – 5	= 100	> 100 + D/2 or < 100 – D/2

- **Time delay.** Specifies a delay between alarm condition occurrence and notification.

Alarm Groups

Alarm notifications will not be received on a ClearView Client PC until the associated Alarm Group has been registered. To register an Alarm Group requires enabling the Alarm Group on the client PC.

Enabling/Disabling Alarm Groups

1. Open Application Explorer.
2. Click **Alarm Groups** under the Setup folder. The Select Alarm Groups dialog box opens.
3. Select/deselect the alarm groups to be notified on the Client PC.
4. Press **Accept** when done to register/unregister the Alarm Groups.

Note: Alarms associated with the selected alarm Groups will become active on the client station; however, deselecting an alarm group requires a restart of the ClearView Client to stop receiving alarm notifications for the alarm group removed.

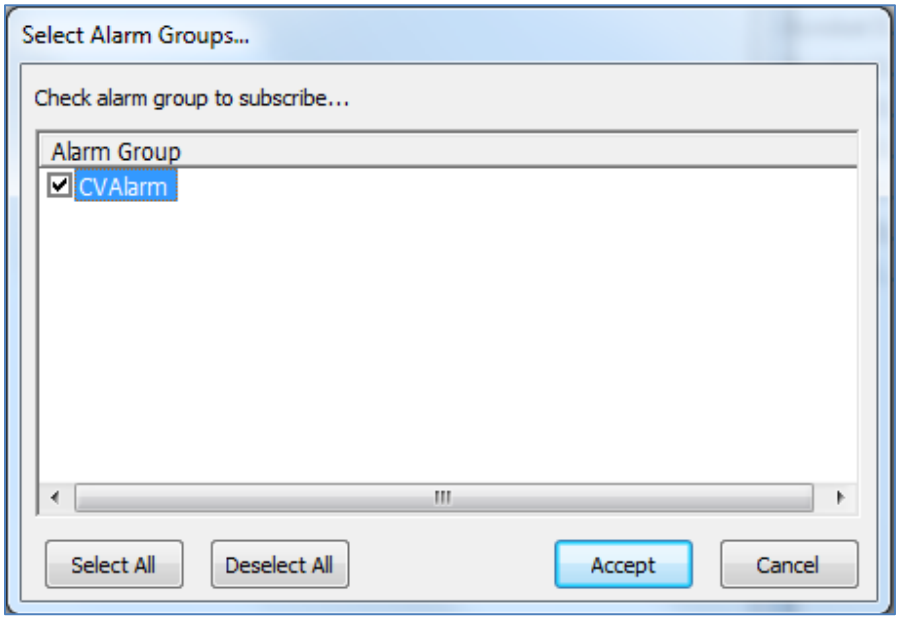


Figure #4.3

Group Alarms

Group alarm is the alarm that represents all alarms in the group it belongs to. Group alarms are calculated and delivered by ClearView-SCADA Server. Group alarms states are calculated based on the states of all (non-group) alarms in the group.

- If one or multiple alarms in an Alarm Group is Active the Group Alarm would receive a status of Active Alarm
- If all alarms in an Alarm Group are Active and Acknowledged the Group Alarm would receive Active and Acknowledged status, otherwise the previous state would remain
- If all alarms in an Alarm Group are Cleared and Unacknowledged the Group Alarm would receive Cleared and Unacknowledged status, otherwise the previous state would remain
- If all alarms in an Alarm Group are Cleared and Acknowledged the Group Alarm would receive Cleared and Acknowledged status, otherwise the previous state would remain
- If all alarms in an Alarm Group are Cleared, Reset and Acknowledged the Group Alarm would receive Cleared, Reset and Acknowledged status, otherwise the previous state would remain

Setting Group Alarm

Check **Group Alarm** on **Alarm Properties** Dialog.

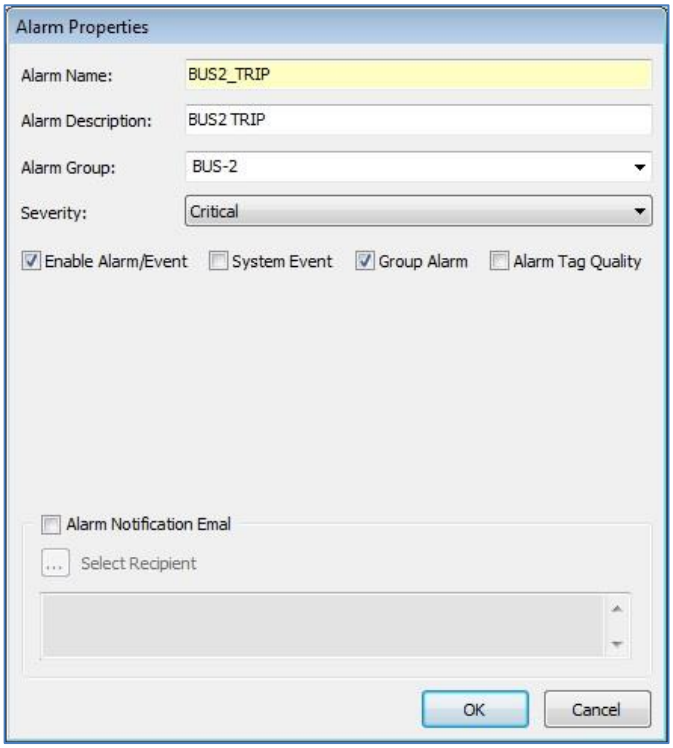


Figure #4.4

Property	Required
Alarm Name	Yes
Alarm Description	No
Alarm Group	Yes
Severity	Yes
Enable Alarm	No
System Event	No
Email	No

Note: It is recommended to have only one Group Alarm per an Alarm Group. In case if user creates more than one Group Alarm per a group the following Message Box will be shown.

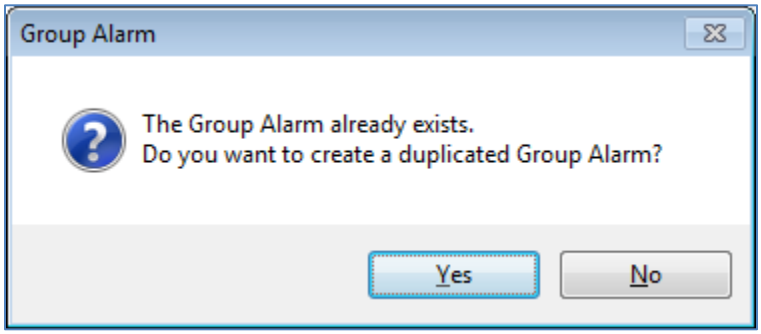


Figure #4.5

Alarm/Event Viewer

The alarm event viewer is a built-in window displaying historical and/or current alarms and security & events logs. To open the Alarm Event Viewer:

1. Open Application Explorer.
2. Expand the Alarms and Events folder.
3. Double-click Alarm/Event Viewer.

Note: The Alarm/Event Viewer is also accessible from the Project menu:

- Choose Alarm & Events and then click Alarm & Event View.

Alarm/Event Viewer Menu

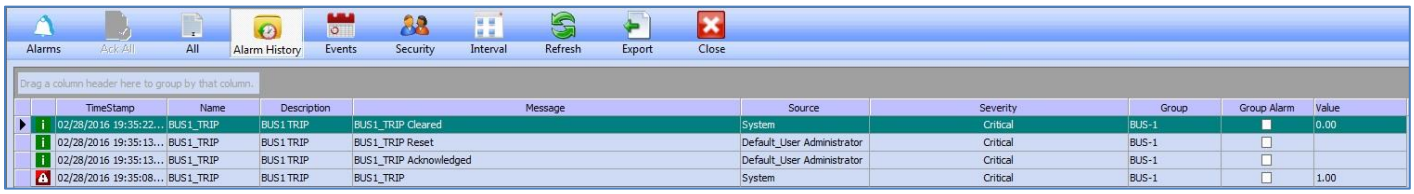


Figure #4.6

Alarms	Displays currently active alarms.
Ack All	Acknowledges all alarms.
All	View all types of alarms logs: alarms, security, and events.
Alarm History	View when alarms have occurred, acknowledged and cleared
Events	View Events only.
Security	View Security Events only.
Interval	Enter and apply a filter to historical events and security listings by a date interval.
Refresh	Refreshes the log data displayed.
Close	Closes the Alarm/Event Viewer window.

Alarm Banner ActiveX Control

A special ActiveX control was developed for ClearView screens. It provides a custom interface for displaying and acknowledging active alarms.

To add the Alarm Banner to a screen:

1. In Design mode, click **Data Controls** in the Object Library, and then click **Alarm Banner**.

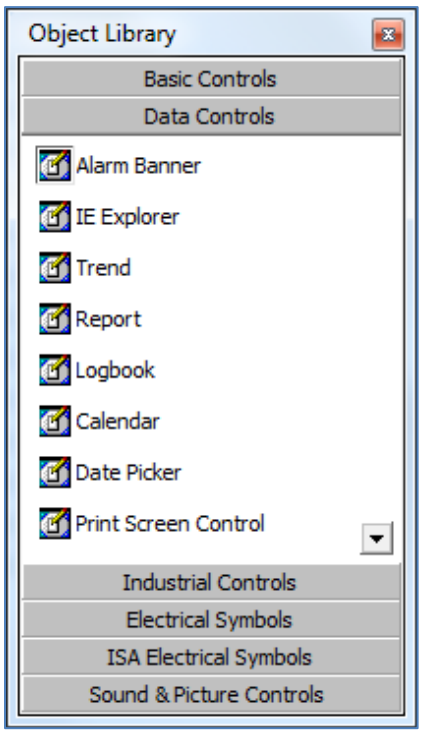


Figure #4.7

2. Now click the Drawing Pad (screen) area.
3. Resize the control to a desired size.
4. Right-click the object and select **Properties ccAlmBanner.ccAlmBan Object** properties.
5. Select and configure the Alarm Banner control using the properties under the General, Columns, Extended Properties property tabs.
6. When done press **Accept**.

Tip: Alarms can be acknowledged individually or collectively using the Alarm Banner control. However, if only certain users are to be allowed to acknowledge alarms, then turn off the Alarm Banner's **User Can Acknowledge Alarm** property. To set individual user's ACK permissions see ClearView Security SECTION 5 below.

Alarm Banner Example

1. Select Alarm Banner from the Object Library and draw it on a ClearView screen.

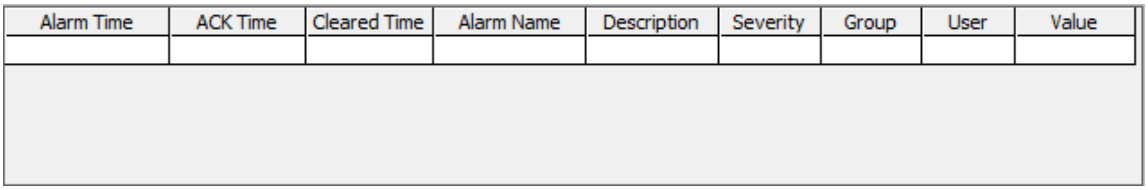


Figure #4.8

2. Click on the object and configure the properties.

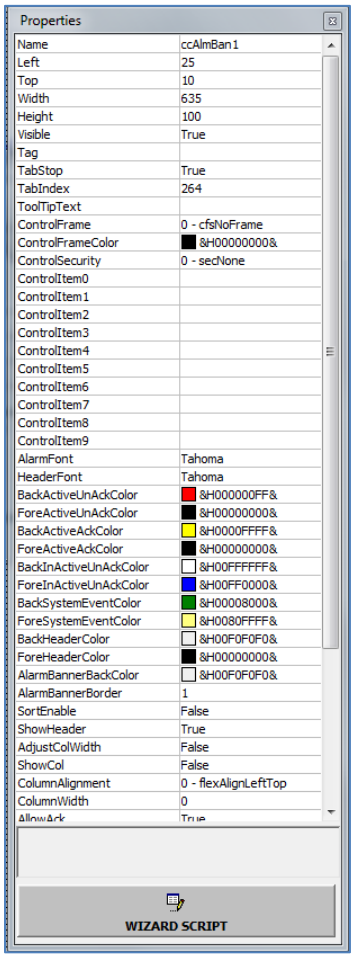


Figure #4.9

3. Add resizing scripting code:

CCAlarmBan1.ColumnWidth (1) = 2300
CCAlarmBan1.ColumnWidth (2) = 2300
CCAlarmBan1.ColumnWidth (3) = 2300
CCAlarmBan1.ColumnWidth (4) = 2300
CCAlarmBan1.ColumnWidth (5) = 3500
CCAlarmBan1.ColumnWidth (6) = 1000
CCAlarmBan1.ColumnWidth (7) = 1000

Note: Resizing the alarm banner must be done in scripting to maintain the desired dimensions.

Setting Default Colors

The default color settings for the Alarm Banner control are:

- Active Alarms – Red background, Black text.
- Acknowledged Alarms – Yellow background, Black text.
- Cleared and unacknowledged Alarms – Blue background, Black text.
- Not reset Alarms – White background, Black text.

Alarm Banner Design Properties

Field	Description
General	
Show Header	Select to display the names of the column headers.
Beep New Alarms	Select to enable audible beep sound on PC.
User Can Acknowledge Alarm	Select to allow any user the ability to acknowledge alarms through the Alarm Banner Control.
Filter Group Alarms (Only)	If checked then the control will display only Group Alarms.
Enable Sorting by Column	Select to allow any user the ability to sort Alarm Banner by any column. Sorting can be descending or ascending.
Blink Unacknowledged Alarms	Select to blink unacknowledged alarms.
Enable Adjust Column Width	Select to allow any user the ability to manually adjust the column widths.
Acknowledge Alarm Group	If checked then a user can acknowledge an entire Alarm Group at once.
Displayed Date Format	Enter the date time format. Example: mm/dd/yy hh:mm:ss am/pm. You can enter “c” for common format.
Columns	
Visibility	Check to show each selected column.
Alignment	Select from dropdown list the text alignment desired.
Width	Pixel width of each column.

Alarm Banner Run-Time (Scripting) Properties

Property	Description	Value/Parameters	Example
AckGroupAlarms	Acknowledge Group Alarm	FALSE/TRUE	AlmBan1.AckGroupAlarms = True
ACKPermitted	User Acknowledge Permitted	FALSE/TRUE	AlmBan1.ACKPermitted = True
AdjustColWidth	Enable/Disable adjustment of the column width in runtime	FALSE/TRUE	AlmBan1.AdjustColWidth = True
AlarmBannerBackColor	Set/Get Alarm Banner Back color	OLE Color	AlmBan1.AlarmBannerBackColor = VBRed
AlarmBannerBorder	Set/Get Alarm Banner Border	Integer 0 and 1	AlmBan1.AlarmBannerBorder = 1
AlarmFont	Set/Get Alarm Banner Font	OLE Font	AlmBan1.AlarmFont.Name = "Tahoma"
AlmAckDateFormat	Set/Get Acknowledged Alarm Date Format	String format value	AlmBan1.AlmAckDateFormat = "dd/mm/yy hh:mm:ss:ll"
AlmActiveDateFormat	Set/Get Active Alarm Date Format	String format value	AlmBan1.AlmActiveDateFormat = "dd/mm/yy hh:mm:ss:ll"
AlmClearedUnAckDateFormat	Set/Get Cleared & Unacknowledged Alarm Date Format	String format value	AlmBan1.AlmClearedUnAckDateFormat = "dd/mm/yy hh:mm:ss:ll"
AutoResetClearedAckAlarms	Set/Get Automatic reset of cleared alarms	FALSE/TRUE	AlmBan1.AutoResetClearedAckAlarms = False
BackActiveAckColor	Set/Get Back Color of Active & Acknowledged Alarm	OLE Color	AlmBan1.BackActiveAckColor = vbRed
BackActiveUnAckColor	Set/Get Back Color of Active & Unacknowledged Alarm	OLE Color	AlmBan1.BackActiveUnAckColor = vbYellow
BackHeaderColor	Set/Get Back Color of the alarm banner header	OLE Color	AlmBan1.BackHeaderColor = vbBlack
BackInActiveUnAckColor	Set/Get Back Color of Cleared & Unacknowledged Alarm	OLE Color	AlmBan1.BackInActiveUnAckColor = vbBlue

Property	Description	Value/Parameters	Example
BackSystemEventColor	Set/Get Back Color of System Event	OLE Color	AlmBan1.BackSystemEventColor = vbBlue
BeepNewAlarm	Enables/Disables sound on the New Alarm	FALSE/TRUE	AlmBan1.BeepNewAlarm = True
BlinkUnAckAlarm	Enables/Disables blinking on the New & Acknowledged Alarm	FALSE/TRUE	AlmBan1.BlinkUnAckAlarm = True
ColumnAlignment	Set/Get Alarm Banner Column Alignment	Column Index & Alignment Value Integer 0-9	AlmBan1.ColumnAlignment(1) = 5
ColumnWidth	Set/Get Alarm Banner Column Width	Column Index & Width Value Long in pixels	AlmBan1.ColumnWidth(3) = 2000
FilterGroupAlarms	Enables/Disables filtering of Group Alarms	FALSE/TRUE	AlmBan1.FilterGroupAlarms = True
ForeActiveAckColor	Set/Get Font Color of Active & Acknowledged Alarm	OLE Color	AlmBan1.ForeActiveAckColor = vbBlack
ForeActiveUnAckColor	Set/Get Font Color of Active & Unacknowledged Alarm	OLE Color	AlmBan1.ForeActiveUnAckColor = vbBlack
ForeHeaderColor	Set/Get Font Color of the alarm banner header	OLE Color	AlmBan1.ForeHeaderColor = vbWhite
ForeInActiveUnAckColor	Set/Get Font Color of Cleared & Unacknowledged Alarm	OLE Color	AlmBan1.ForeInActiveUnAckColor = vbBlack
ForeSystemEventColor	Set/Get Font Color of System Event	OLE Color	AlmBan1.ForeSystemEventColor = vbBlack
ShowCol	Enables/Disables column visibility	Column Index & FALSE/TRUE	AlmBan1.ShowCol(4) = True
ShowHeader	Enables/Disables Alarm Banner Header	FALSE/TRUE	AlmBan1.ShowHeader = False

Property	Description	Value/Parameters	Example
SortEnable	Enables/Disables Alarm Banner Sort	FALSE/TRUE	AlmBan1.SortEnable = True

Alarm Banner Methods

Method	Description	Value/Parameters	Example
AdminAck	Enables/Disables alarm banner context menus	FALSE/TRUE	AlmBan1.AdminAck False
ApplyFilter	Applies Filter by specifed column by a search string(s) seperated by pipe symbol "|"	Column Index & search string	AlarmBanner.ApplyFilter 6, “Priority1|Pririorty10”
RemoveFilter	Removes Column Filtering	None	AlmBan1.RemoveFilter
RemoveFilterGroupAlarm	RemovesGroup Alarm Filtering	None	AlmBan1.RemoveFilterGroupAlarm
SilenceAlarm	Enables/Disables Silencing of the alarm sound	FALSE/TRUE	AlmBan1.SilenceAlarm False

Alarms Set-Point Interface

Alarms Set-Point Interface provides an access to Alarm Data base to enable or disable alarms or change alarms’ set-point values.

To get access to “Alarms Set-Point” interface select from Application Explorer **Setup | Alarm Set-Point** or from application main menu **Project | Setup | Alarm Set-Point**

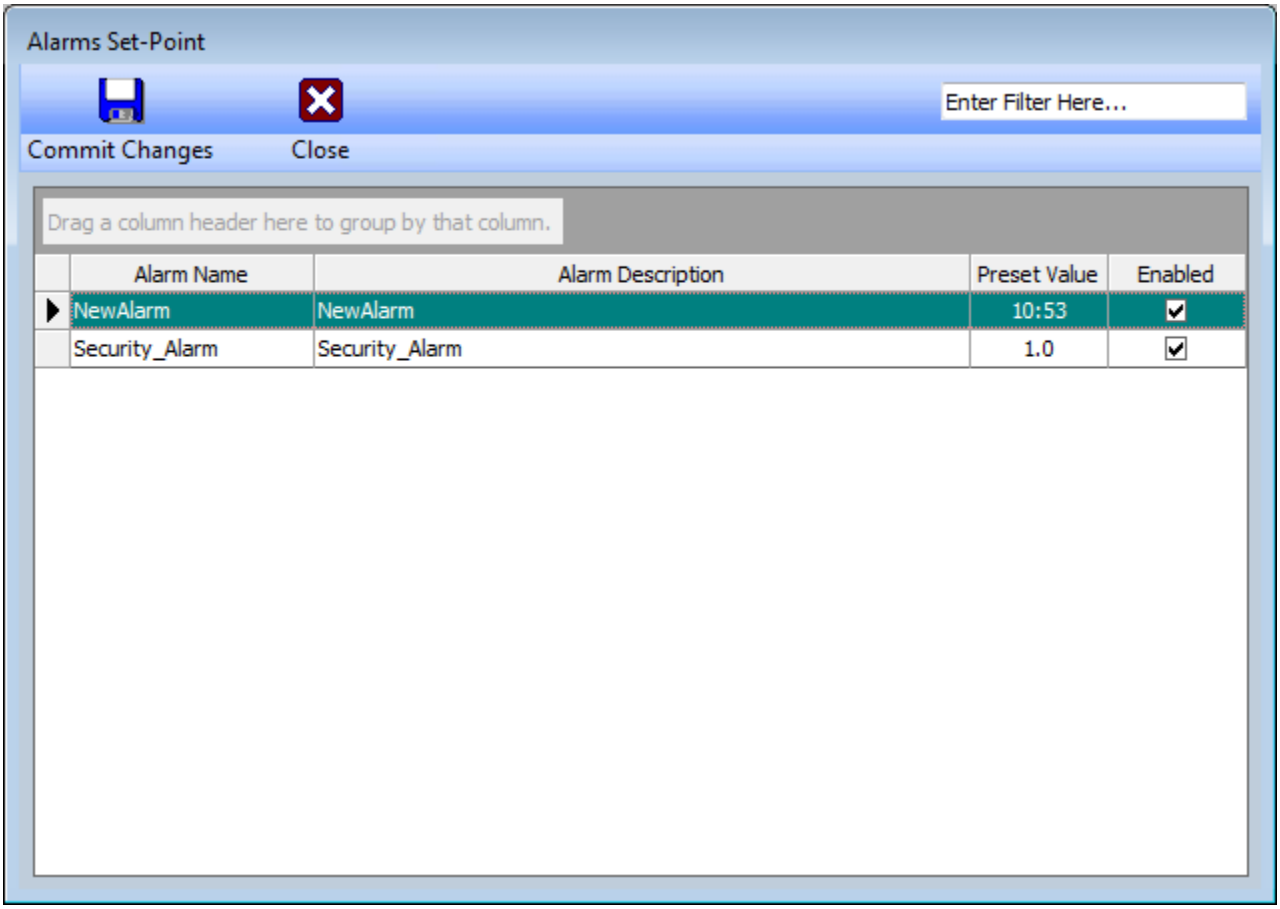


Figure #4.10

- To change Alarm **Preset Value** enter a new value in desired cell
- To enable or disable alarm check or uncheck checkbox for selected alarm in **Enable** column
- To commit changes and exit the interface click on toolbar **Commit Changes** button
- To exit the interface click on toolbar **Close** button

Log Book

The Log Book provides a convenient method for recording notes and setting up follow-up reminders after an alarm condition change has been detected by the operator.

To view the Log Book, navigate the Alarms and Events folder in the Application Explorer and select Log Book.

Making an Entry in the Log Book

1. Click **Edit Log Book** when displayed on the **Status Bar**.

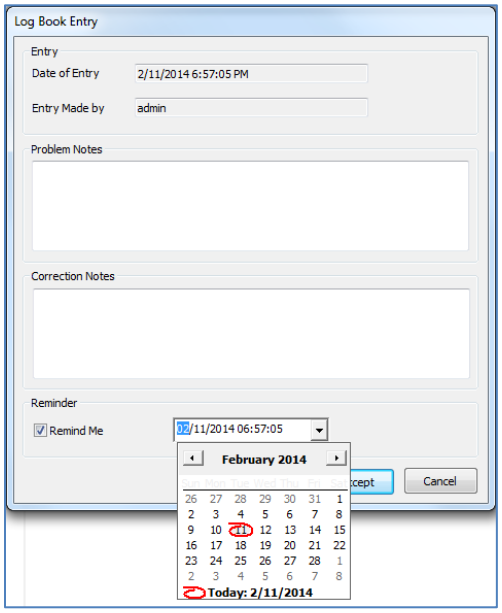


Figure #4.11

2. Fill in the entry fields.
3. Click **Accept** to save and close the entry form.

Tip: The Log Book Entry is also accessible from the Application Explorer under the Alarm and Events folder. Double-click **Log Book** to open the Log Book window to view all past entries. Then press New Entry.

Viewing Past Log Book Entries

The Log Book form includes two calendar controls. Use these controls to specify a range of logs to view.

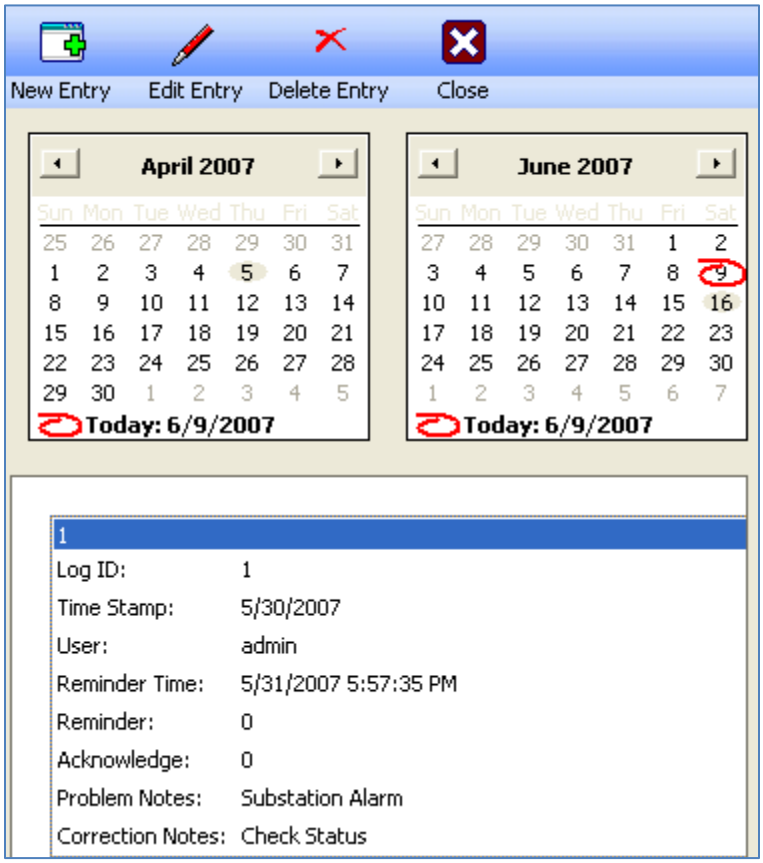


Figure #4.12

Alarm/Event Reports

The Alarm and Event Log Reports are built-in Crystal Reports modified to recover data from the Event_Log database table. They provide detailed reports of when alarm/event conditions change.

How to Open a Log Report

1. Open **Application Explorer**.
2. Expand the **Reports** folder.
3. Expand **Logs** subfolder
4. Expand the subfolder **Alarm and Events**.
5. Double-click **Alarm Log** or **Event Log** to open the corresponding report. For more information on using reports, see SECTION 6 below.

Note: As the database table gets large, the reports will open slower.

Alarm Log Report Fields

Field	Description
Date/Time	Date and time when the alarm condition changed.
Alarm State	Value of the alarm condition.
Alarm Message	Displays the Alarm Description as defined in the Alarm table.
Severity	Displays the Severity as defined in the Alarm table.
User Name	Displays the name of the user who acknowledges the alarm. The other events will display the username as system.

Event Log Report Fields

Field	Description
Date/Time	Date/Time Date and time when the event occurred
Message	Displays a description of the event
User Name	Displays the name of the user associated with the event

Emailing Alarms

Alarm conditions can be transmitted as text messages to cell phones via SMTP server or to an email address over the Internet.

There are three steps to configuring emailing alarms: (1) Configure Project Email, (2) Configure Users and (3) Configure Alarms.

Step 1 – Configure Project Email

Note: Check with your system administrator about configuring your connection to your service provider.

1. On **File** menu, select **Project Settings**.
2. Select the **Email** tab.
3. Enter the name of your company's **Email Server** (mail@sbcglobal.net).
4. Enter the email user account assigned to the ClearView Server (SysAlarms@MyCompany.com)
5. Enter the **Email Subject** for the message (SysAlarm).
6. If your service provider requires a username and password, enter them in the **Email User** and **Email Password** fields.
7. Select the type(s) of alarm notifications to send by checking any of the **Alarm Email Settings**.

Note: Alarms can only be acknowledged on a ClearView Client

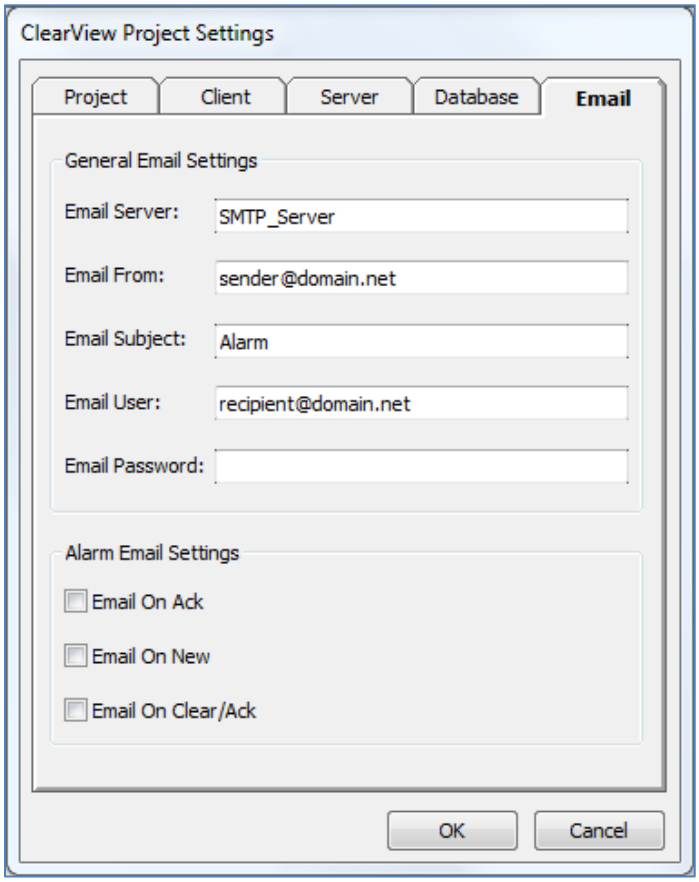


Figure #4.13

Step 2 – Configure Users

1. On **Application Explorer**, select **Users Database** under the **Setup** folder.
2. Add/edit users to receive the alarms to include an email address or cell number by entering the information in the **Email** field.

Tip: Email a message to user Joe Maintenance by entering his email address (JoeMaint@relabsoft.com). Or enter a cell phone number that can receive text messaging (9251234567@mobile.att.net). Where the format is the cell phone number without the dashes (925-123-4567)

Step 3 – Configure Alarms

Add/edit the alarm tags to be broadcast by checking the **E-mail** box in the Alarm configuration dialog box.

1. Click **Select Recipient**. The Choose recipients dialog box opens.
2. Check the names to receive alarm notifications.
3. Click **Accept** when done.
4. Click OK to save the Alarm Properties configuration.

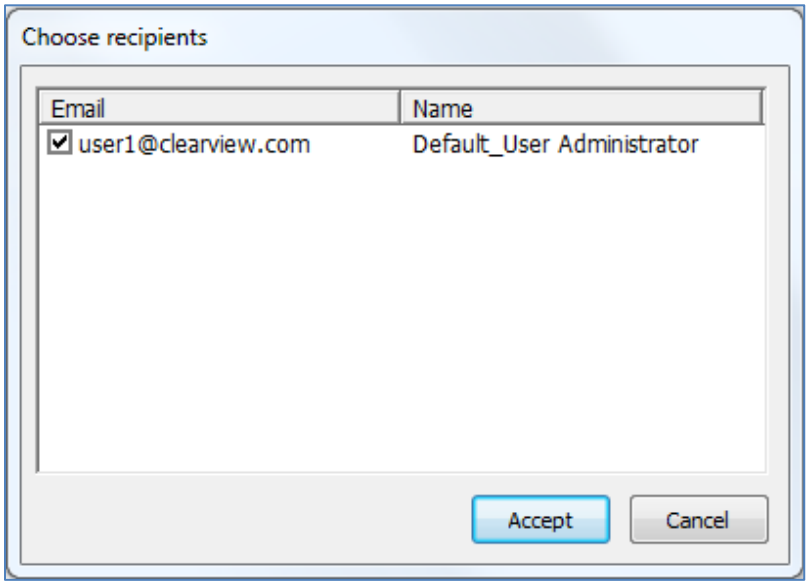


Figure #4.14

Referencing Alarms through Scripting

Below are scripting code examples for use with Alarms/Events.

```
'Opening the Alarm Event Viewer
Screens.ShowScreen "EventView"
'Closing the Alarm Event Viewer
Screens.CloseScreen "EventView"
'To Acknowledge All alarms use the following script code:
SubscribedAlarms.AcknowledgeAll
```

Note: See the Script Reference help in the appendix for detailed information on the SubscribedAlarms Collection.

SECTION 5

Security

ClearView security controls access to the File menu, Application Explorer, Login/Logout and screen graphics. When creating a user it is necessary to assign that user to a group. The group indicates the capabilities the user will have. To fine tune settings for a user, a new group can be created.

In addition to access, ClearView also has built-in 21CFR part 11 requirements. These include tracking tag writes.

User Groups

ClearView security is made up of groups. Each group has permission to do certain tasks. These tasks include menus, forms, reports, etc. It is possible to create a new group and customize the permissions for the group.

A group has a set of permissions. Each user must belong to a group. The default groups that already exist in a ClearView database are:

- Administrator
- Operator
- Supervisor
- View Only

You access the User Groups dialog box by clicking Setup Groups on the User Form toolbar.

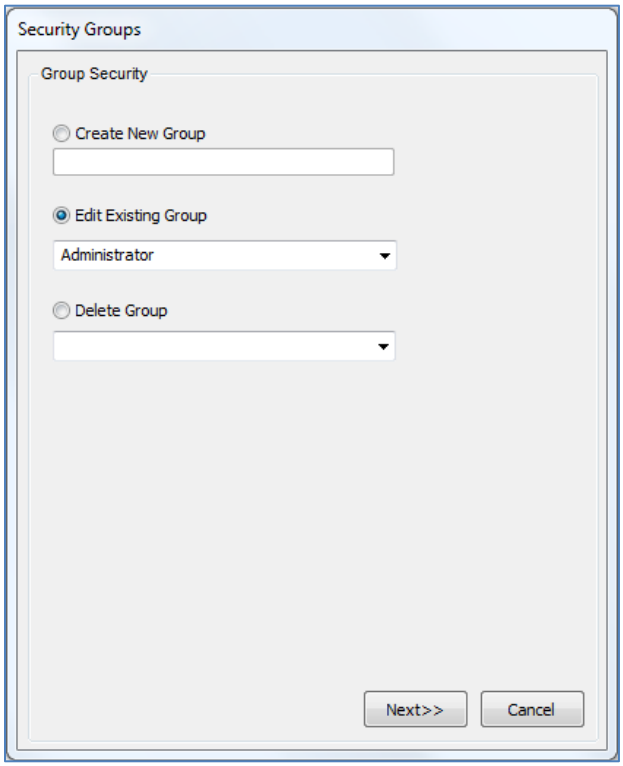


Figure #5.2

Add a New Group

1. Select **Create New Group** option and specify a new group name.
2. Click **Next**.
3. Specify which permissions to give to this group.
4. Click **Accept**.
5. Click **Cancel** when done.
6. Select a user on the Users form and specify this new Group for that user.

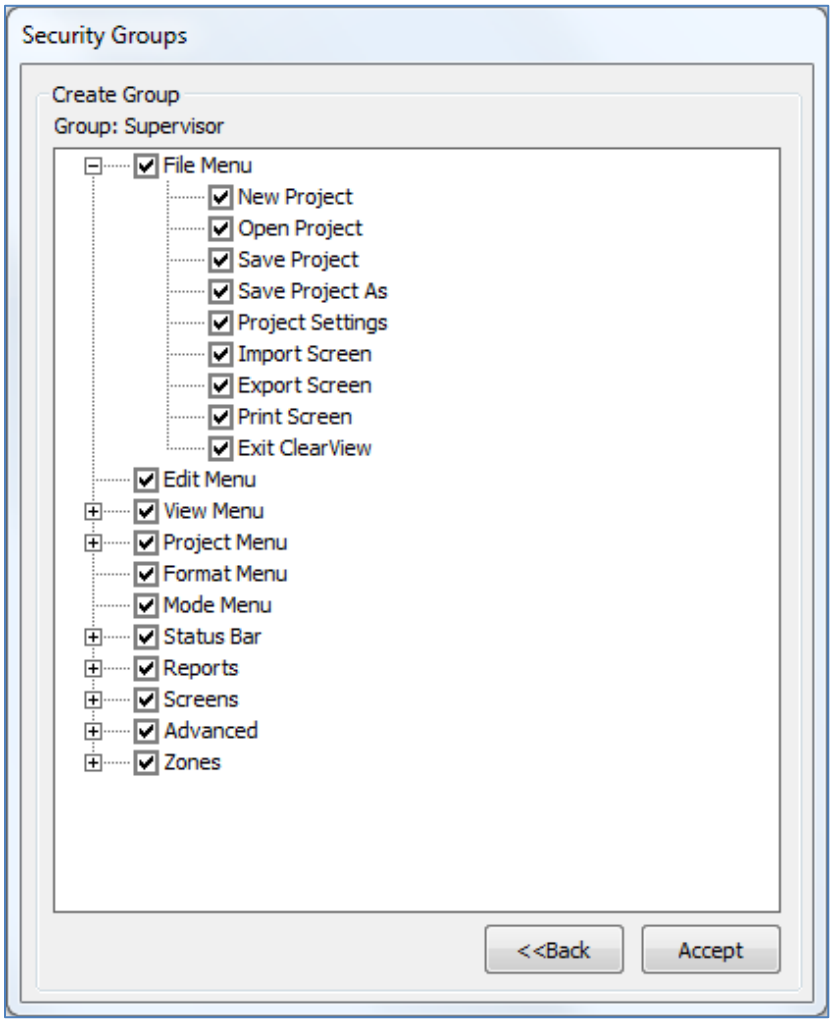


Figure #5.3

Modify an Existing Group

1. Select **Edit Existing Group** option.
2. Select a group from the drop-down menu.
3. Click **Next**.

4. Specify which permissions to give to this group.
5. Click Accept.

Delete Existing Group

1. Select **Delete Group** option
2. Click **Next.**
3. Click **Accept.**

WARNING: Do not delete the Administrator group.

Users

To open the Users Form:

- Go to Application Explorer and select the Users Database icon from the Setup folder

This form is an interface to the ClearView Users database table. It contains application access security information, telephone numbers and e-mail addresses. All fields displayed can be modified. Changes take effect immediately.

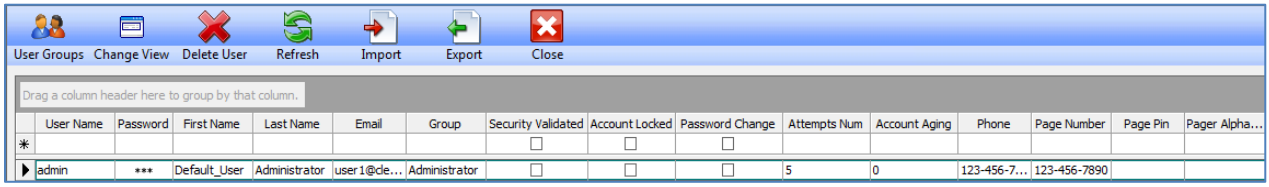


Figure #5.1

The User Form Toolbar

Item	Description
User Groups	Displays Groups dialog box. Create new, edit existing and delete existing groups.
Change View	Toggles between list and tile views.
Delete User	Deletes selected user.
Refresh	Refreshes view.
Import	Exports users database to a CSV file
Export	Imports users from a CSV file.
Close	Closes User Form.

Create a New User

To add a user,

1. Fill in the blank row at the top of the table (marked with an asterisk).
2. Specify the following settings for the new user:
   - **Username**: A 50-character string field containing a unique user name. Generally the user name is constructed from the user's first initial and last name. The user name is entered in the User Name text box on the Login form and appears in the status bar on the bottom of the ClearView screen.
   - **Password**: A 50-character string field containing a unique user password
   - **Last Name**: A 50-character string field containing the user's last name
   - **First Name**: A 50-character string field containing the user's first name
   - **Email**: A 50-character string field containing the user's e-mail address
   - **Telephone**: A 50-character string field containing the user's telephone number
   - **Group** – A group determines the permissions for this user
   - **PageNum** – Pager number.
   - **PagePin** – Pin number.
   - **AccountLocked** – Lock account for this user.
   - **AccountAging** – Specify days that the password will be valid.
   - **Number attempts** – during login a user can only specify these many attempts.
   - **Force Password Change** – At next login the user will have to change password.
   - **Security validated** – Enable security features. See below.

Security Features on the Users Form

By checking the **Security Validated** box on the Users Form, the following features are enabled:

- **Account aging**: Sets the number of days before a password change is required. For example, setting Account Aging to 30 would require the user to change passwords every 30 days. An administrator can force a user to change passwords at any time by checking the Force Password Change box. Setting Account Aging to 0 disables this feature, and the user's password will never expire.
- **Number of Attempts**: Sets the number of invalid password tries before the user's account is locked. For example, if the Number of Attempts is set at 3, and a user fails to enter the proper password 3 times in a row, the account will be locked and the user will be unable to log in. An administrator can unlock the account by un-checking the Account Locked box. An administrator may also lock out an account at any time by checking the Account Locked box. If Number of Attempts is set to 0 this feature is disabled and the account will not lock due to invalid attempts.
- **Changing Passwords (click Password column):** An administrator may reset a user's password by clicking the password box and entering a new password. This will automatically set the Force Password Change box, and the user will be required to change the password when logging in. A user's password must be at least 6 characters long. The password must be different then the username, and cannot be the same as the previous password.

Delete User

To delete a user, select a record and press the **Delete** key on the keyboard.

Scripting

Sometimes it may be necessary to use security in scripting. The following may be accomplished via scripting:

- Block users from gaining access to Menus and Forms.
- Specify zones for controls on a ClearView screen.
- Automatically login a user.

Scripting Examples

To block a user from gaining access to menus and forms:

- Create a new security group that does not include the prohibited menus, forms, toolbars, etc. There is no scripting involved for this task. When this user logs in the security settings will take effect.

To restrict access to certain controls on a screen:

1. Use the ControlSecurity property of a control to specify a zone. For this example select 1 – Zone A.

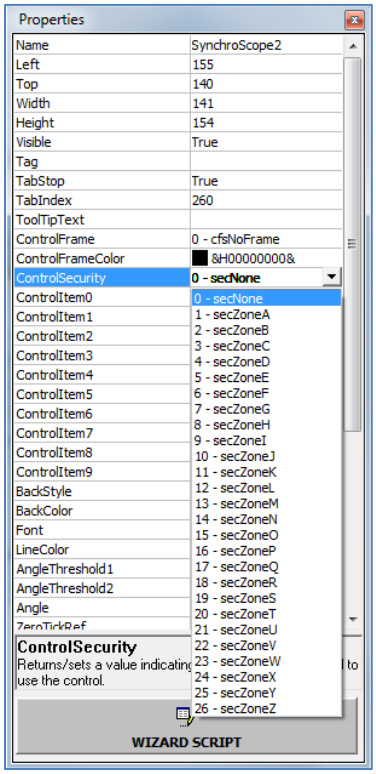


Figure #5.4

2. Next create a group that does not allow access to 1- Zone A.
3. Enter the following script on the Click event of that control:

```
Private Sub ButtonGraphicalCtrl3_Click()
    If Security.Check("1",,0) = True Then
            "Your Code Here"
    End If
End Sub
```

4. Assign a user to the newly created group.

5. After the user logs into ClearView they will receive an “Insufficient security” message box upon clicking the control.

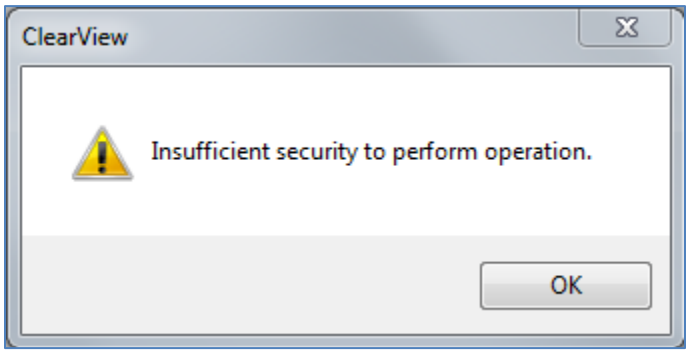


Figure #5.5

Logging In/Out

On the main ClearView toolbar. Click **Login**

Note: If a user is already logged in, you will be prompted to logout

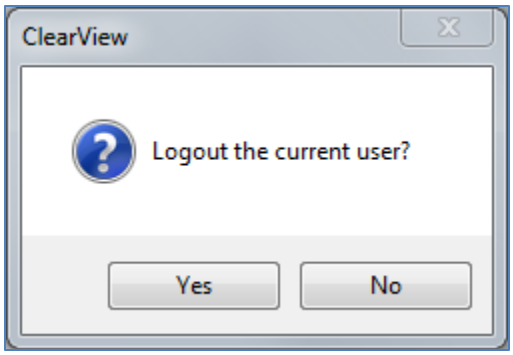


Figure #5.6

- To logout, click the Toolbar Logout button.
- To login or logout via scripting, use the Security object. For example, you can use the following functions:

Security.Login	This will display the security login box
Security.Login “user”, “pass”	This will login the specified user
Security.Logout	This will logout the current user
MsgBox Security.CurrentUser	This will display who is currently logged in

Setting Up Auto Login User

In Project Settings on the Client tab, check the **Auto Login to User** option and specify which user to login.

ClearView-SCADA Security Module

ClearView-SCADA Security Interface to Active Directory

The ClearView Security Interface is a set of functions and methods that allow logging to ClearView based on the list of users in Domain Active Directory even if such user does not exists in ClearView Users database.

ClearView Security Interface allows a user to use network domain credentials to log into ClearView.

The Interface provides functions that can be used in ClearView script to access Active Directory. The functions allow to check if the user exists in a domain, to retrieve user's first, last, middle name and group, and to add the user to ClearView database.

Note: To make this module work properly, a domain user must be assigned to just one group on the list of user's groups in the Domain Active Directory. These user's group names must be identical to the names of groups that were created in ClearView SCADA Security.

Security module functions.

Function	Description	Parameters
Security.VerifyUser(UserName)	Verifies if the user exists in ClearView database. Function returns TRUE if user exists and FALSE if the user doesn't.	UserName – User Name
Security.Export(path)	Exports users from the specified path (.csv file). Returns TRUE if export was successful and FALSE if export fails.	Path – path and file name to .csv file
Security.ImportUsers(path)	Imports users from specified path (.csv file). Returns TRUE if import was successful and FALSE if import fails.	Path – path and file name to .csv file
Security.UnsecureLogin(UserName)	Performs login to ClearView based on provided user name. Password is retrieved from the database. If Login is successful, the function would return TRUE; otherwise returns FALSE.	UserName – User Name
Security.GetCurrentUserGroup	Function retrieves a group to which the current login user belongs to and returns the group name	No parameters
Application.GetWindowsLoginUser	Function retrieves the user currently logged in in to Windows OS and returns the name of login user	No parameters
Security.DomainUserExists(UserName, domain, group)	The function checks if the user belongs to any specified groups in Active Directory. If yes the function returns TRUE. If not function returns FALSE. After the function	UserName – User Name Domain – Domain Name

	is executed the property credentials are populated - DomainUserFirstName - DomainUserMiddleName - DomainUserLastName - DomainUserGroup	Group – Array of Group Names
Project. AddDomainUsers(UserName, FirstName, LastName, UserGroup)	The function adds a new user to ClearView-SCADA database. If the user was successfully added the function returns TRUE, otherwise the function would return FALSE Note: Encrypted password is automatically created which is comprised of prefix "CV_" and User Name	UserName – User Name FirstName – FirstName LastName – Last Name User Group – ClearView-SCADA configured user group

Security module methods.

Method	Description	Parameters
Application.Windows_LogOff	Method performs User Windows Logoff	No parameters

Security module events.

Event	Description	Parameters
Security_UserDBChanged(User)	ClearView-SCADA security module will raise an event if any user(s) credentials are changed	User – User Name

Security object properties.

Property	Description
Security.DomainUserFirstName	Property returns user's first name after successful execution of DomainUserExists function. If not, an empty string would be returned.
Security.DomainUserMiddleName	Property returns user middle name after successful execution of DomainUserExists function. If not, empty string would be returned.
Security.DomainUserLastName	Property returns user last name after successful execution of DomainUserExists function. If not, empty string would be returned.
Security.DomainUserGroup	Property returns to which group the user belongs to after successful execution of DomainUserExists function. If not, empty string would be returned.

Sample Script of Initial Log in to ClearView

This script provides an example of how the login stage can be controlled. There are many other ways to do it, depending on your needs.

```
Call DomainLogin

Note: The default user password that is automatically created by Project.AddDomainUsers would be combined of
"CV_" & "User

Name" provided by Application.GetWindowsLoginUser


Private Sub DomainLogin()
 Dim MN
 Dim n(3) ' array of groups which are present in ClearView as well as in Domain Active Directory
 Dim tmpX ' variable verifies if the user exists in Domain Active Directory database
 Dim tmpY ' variable verifies if the user was created already in ClearView database
 If Security.CurrentUser = Application.GetWindowsLoginUser Then
  Exit Sub
 End If
 n(0) = "CV_Eng"   ' Engineer Group
 n(1) = "CV_Oper"  ' Operator Group
 n(2) = "CV_Admin" ' Administrator Group
'Verifies if the user already exists in ClearView database and if YES performs Login based on currently logged in
domain user
 If Security.VerifyUser(Application.GetWindowsLoginUser) Then
'ClearView logout
  Security.LogOut
'ClearView Login based on retrieved password for user to login
  Security.Login Application.GetWindowsLoginUser, Security.UnsecureLogin (Application.GetWindowsLoginUser)
  Exit Sub
 Else


  'If the user doesn't exist in ClearView database the Sub would check if the user exists in Domain Active Directory
   tmpX = Security.DomainUserExists (Application.GetWindowsLoginUser, "relabsoft", n)
```

```
'If the user exists in Domain Active Directory the Sub will automatically add a new user based on information retrieved from
'the database, with the password "CV_" & "User Name" 'First Name, Last Name and User Group
 If tmpX Then
  If Security.DomainUserFirstName = "" Or Security.DomainUserLastName = "" Or Security.DomainUserGroup = "" Then
    Application.MsgBox "Domain User is not properly configured"
    Security.LogOut
    Security.Login
    Exit Sub
  End If
 If Security.DomainUserMiddleName <> "" Then 'Retrieving Middle Name,
  MN = " " & Security.DomainUserMiddleName
 Else
  MN = ""
 End If
tmpY = Project.AddDomainUsers (Application.GetWindowsLoginUser, Security.DomainUserFirstName & MN,
Security.DomainUserLastName, Security.DomainUserGroup)
 End If

'If operation to add user was successful then the user would be logged in automatically
 If tmpY Then
  If Security.UnsecureLogin (Application.GetWindowsLoginUser) <> "" Then
     Security.LogOut 'ClearView logout
'ClearView Login based on retrieved password to allow user to login
     Security.Login Application.GetWindowsLoginUser, Security.UnsecureLogin (Application.GetWindowsLoginUser)
  End If
 End If
 End If
End Sub
```

Active Directory Login (DomainLogin Method)

Security.DomainLogin(DomainName, UserGroups) can be called from ClearView-SCADA scripting. Two parameters are specified in calling this method:

- **DomainName** is the name of the domain where user will be logging to.
- **UserGroups** is the name of groups that are configured in Active Directory and ClearView-SCADA security. The number of groups that can be specified is unlimited and the groups should be separated by comma ",".

Example: Security.DomainLogin "MyDomainName", "Group1,Group2,Group3,etc"

As soon as the method is executed, the following interface will appear. (See Figure #5.7.) The user will be prompted to enter Active Directory User Name and Password. Pressing **OK** will perform user validation against Active Directory.

Figure #5.7

If such user is validated but did not exists in Clear View Users Database the user will be added to the database with the password "CV_"&"User Name from Active Directory", see Figure # 5.8 below.

Operational Functionality

The diagram below shows the operational flow of ClearView Security Module Interface to Active Directory.

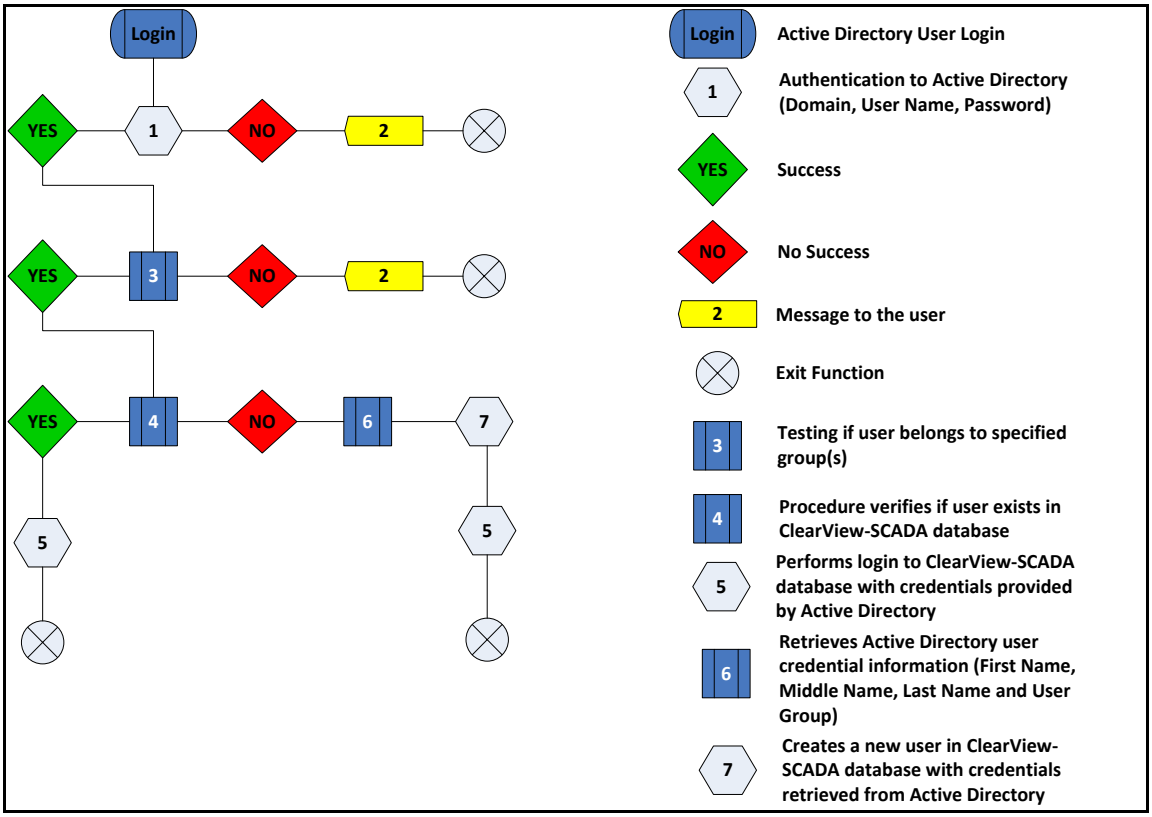


Figure #5.8

SECTION 6

Reports

ClearView generates reports from the data logged or stored in the database tables via a Crystal Reports Interface. Reports can be filtered (selected), sorted, previewed, printed, or exported.

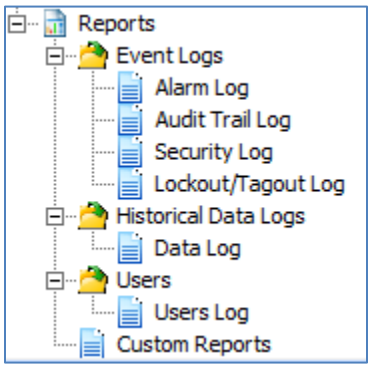
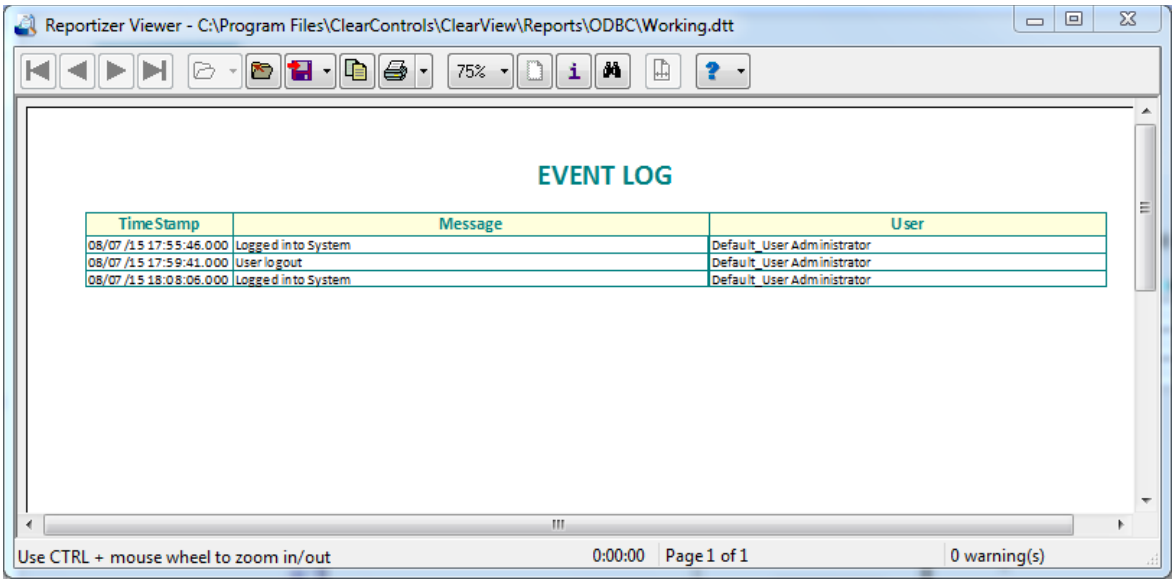


Figure #6.1

Alarms log, Audit Trail Log, Security Log, Lockout/Tagout , Data Log and Users Log are accessed from the Application Explorer (Figure #6.1). In addition, a browser dialog box can be used to open any Custom Reports stored within your network.

Reports Interface

Using the Application Explorer, double-click a report in the Reports folder. The “Reportizer Viewer” Interface will be displayed (Figure #6.2). Please see the “ClearViewReports“ for further details.



Time Stamp	Message	User
08/07 /15 17:55:46.000	Logged into System	Default_User Administrator
08/07 /15 17:59:41.000	User logout	Default_User Administrator
08/07 /15 18:08:06.000	Logged into System	Default_User Administrator

Figure #6.2

Built-In Reports

To open a built-in report:

1. Open ClearView **Application Explorer.**
2. Expand the Report folder/subfolder (Figure #6.1).
3. Double-click the required report.

Report Settings Dialog

To customize built-in reports right-click on the report; the following context menu will appear.

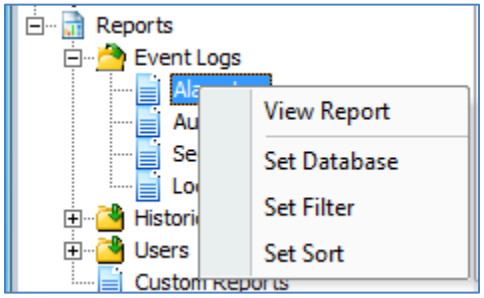


Figure #6.3

Clicking on Set Database, Set Filter or Set Sort will open the Report Settings Dialog.

Using Report Settings Dialog you can:

1. Select previously configured DSN on Database tab.

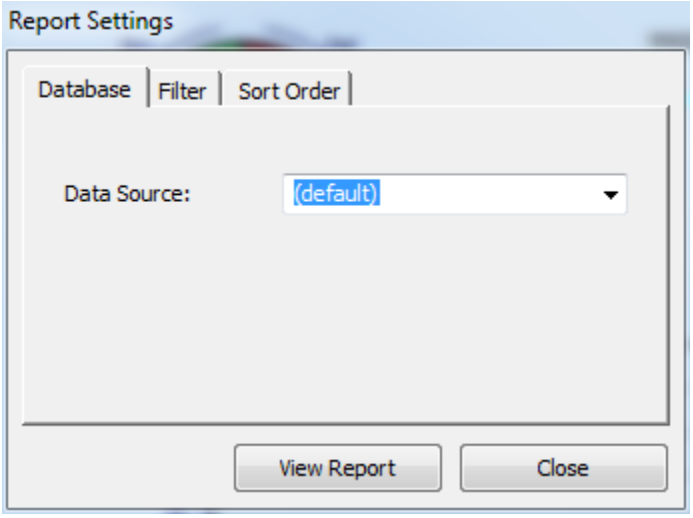


Figure #6.4

2. Filter data by Start Time and End time using Filter Tab

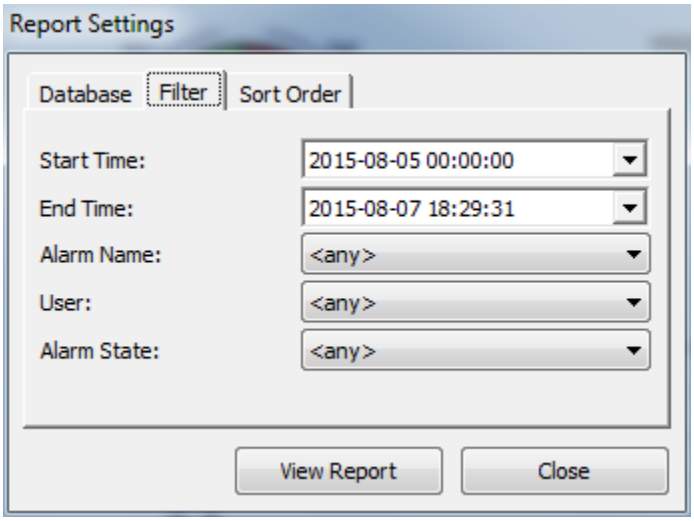


Figure #6.5

3. Specify Timestamp sorting order on Sort Order Tab.

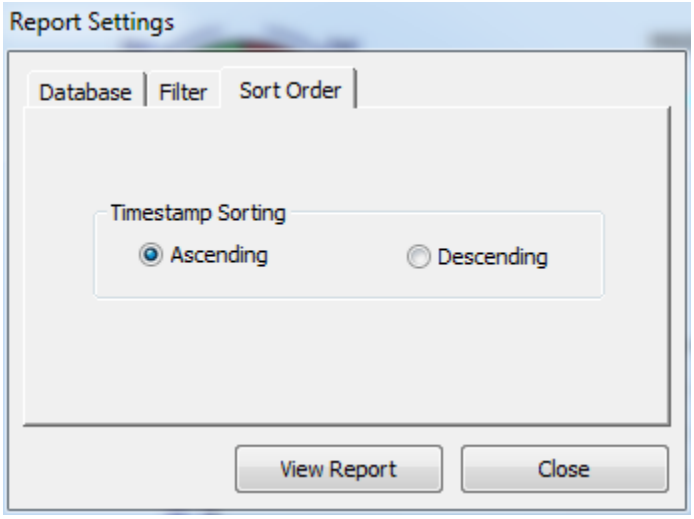
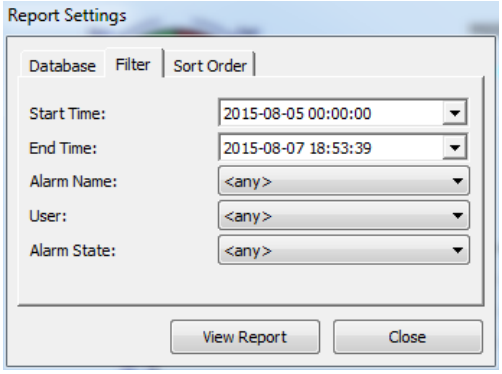
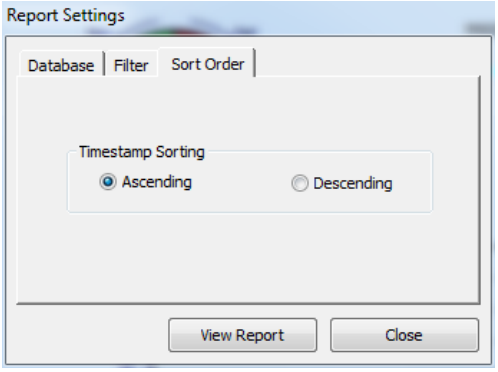
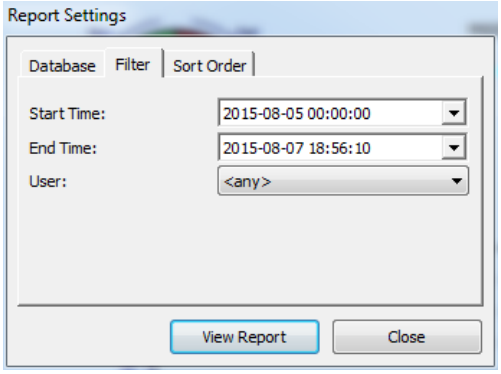
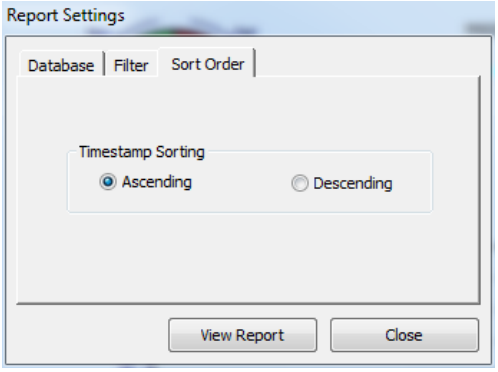
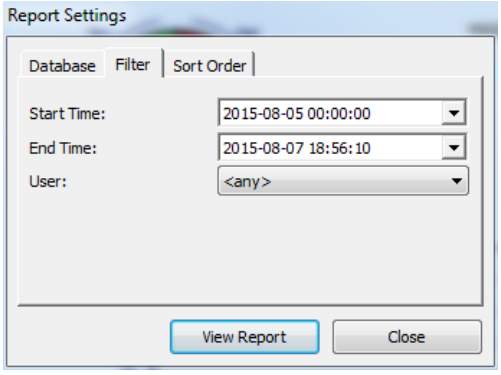
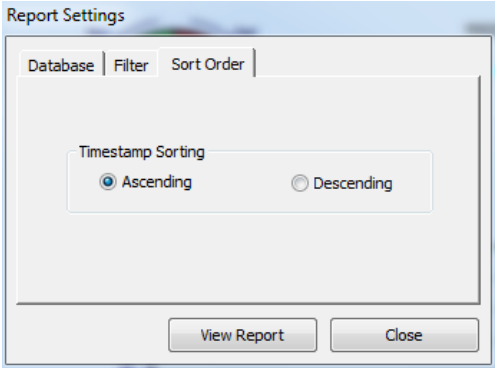
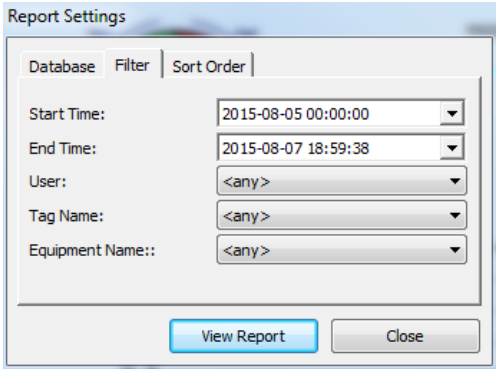
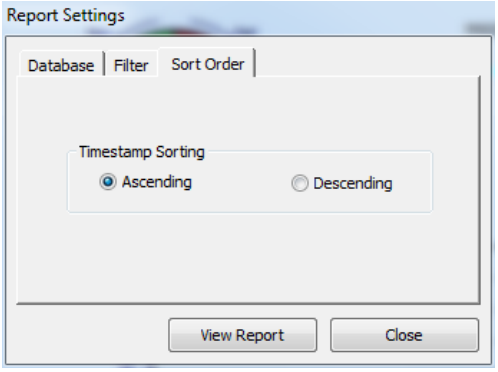
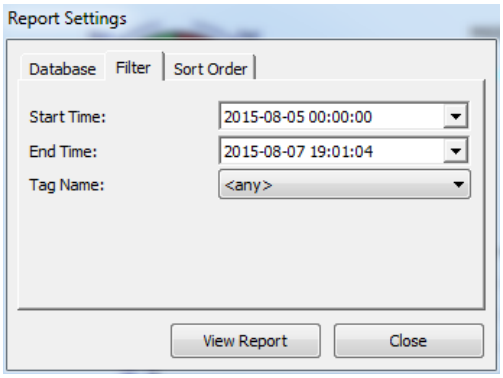
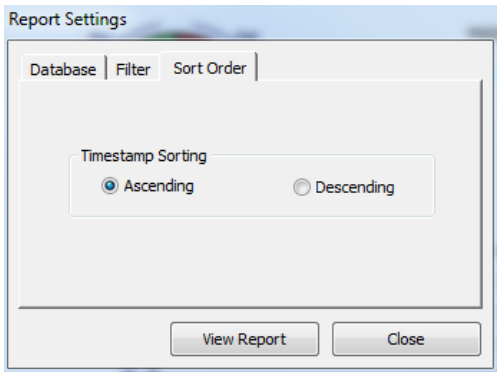


Figure #6.6

Press **View Report** or **Close** when done.

Filters and Sort Order per built-in report

Build-in report	Filters	Sort Order
Alarm log		
Audit Trail Log		
Security Log		
Lockout/Tagout Log		

Data Log		
Users Log	N/A	N/A

Note: Opening a report may take excessive time if the database table is very large. Database management should be implemented to minimize the active tables, back up the data tables, and purge old data on a daily basis.

Note: The filter settings selected on a report are retained even when a project is closed and reopened.

Opening Reports from ClearView script

Built-in and custom-designed reports can also be accessed using ClearView scripting.

To open a report, use the **ViewReport** method of the **Project** object

Project.ViewReport "Report Name"

Examples:

```
Project.ViewReport "Alarm Log"
Project.ViewReport "Audit Trail Log"
Project.ViewReport "Security Log"
Project.ViewReport "Lockout/Tagout Log"
Project.ViewReport "Data Log"
Project.ViewReport "Users Log"
Project.ViewReport "Custom Reports"
```

Opening “Custom Reports” from a script will show a Windows file explorer. User can browse for and open a particular report file.

SECTION 7

Tools

SQL Expression Builder

The SQL Builder is a tool to build/execute simple SQL statements. Depending on the statement built, a return value will be displayed. SQL expressions can also be imported or exported to/from a saved file.

WARNING: The DELETE, UPDATE and INSERT INTO expressions will alter a database. Therefore, security access to the SQL Builder should be limited.

The SQL Expression Builder allows a user to build an executable SQL statement using the following format:

ExecuteSQL(<select database>; <SQL command>)

The **ACCEPT** button becomes available when a valid SQL command is properly built. Press **Clear**, **Entry,** or **Cancel** to start over.

You can build SQL commands using the following formats:

SELECT *<select field>* FROM *<select table>* WHERE *<select field>* = [*enter expression*] ORDER BY *<select field>* DESC/ASC
DELETE FROM *<select table>* WHERE *<select field>* = [*enter expression*] ORDER BY *<select field>* DESC/ASC
UPDATE *<select table>* SET *<select field>* = [*enter expression*] WHERE *<select field>* = [*enter expression*] ORDER BY *<select field>* DESC/ASC
INSERT INTO *<select table>* (*<field 1>*, *<field 2>*, … *<field N>*) VALUES ('[*value 1*]', '[*value 2*]', … '[*value N*]') **Note**: The *<select field>* or *<select table>* provide valid fieldname and tablename selections. For the [*enter expression*], right-click to select from a popup menu for clearing the entry, entering your own string or numeric value, or selecting a Tagname or Value from the current ClearControls database.

Backup Files Function

The ClearView program provides a Backup function that can help a program user to save and store files and information. This Backup function is for files in the project not the project as a whole. The Backup function is accessible from either the Project menu's Setup submenu or in the Application Explorer in the Setup folder.

The Backup function allows users to select files to be stored and select a time interval or frequency event.

Backup Control

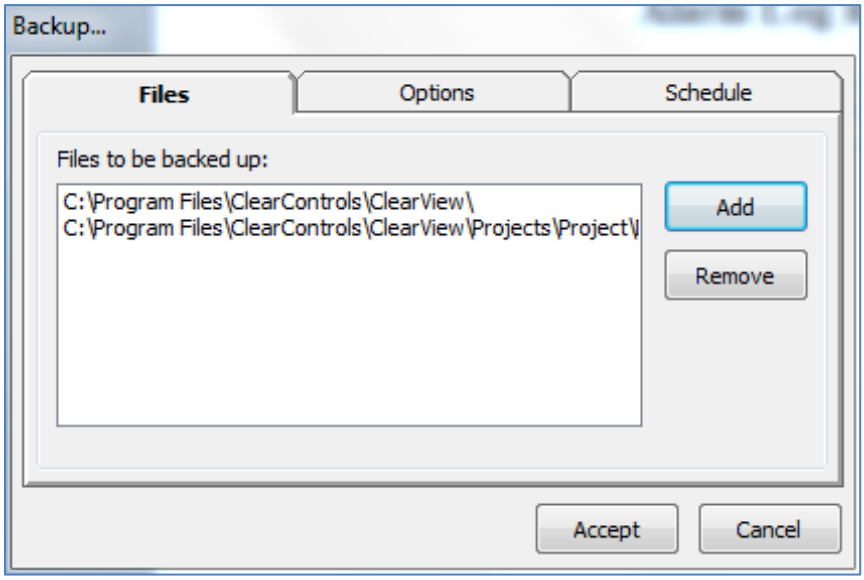


Figure #7.1

Files

The Files tab shows the list of files a user has selected for Backup. To select files and edit the list of Files to be backed up the user should use the Add and Remove button.

Add

When the user selects **Add**, the Select window is displayed and the user can then select which files will be backed up.

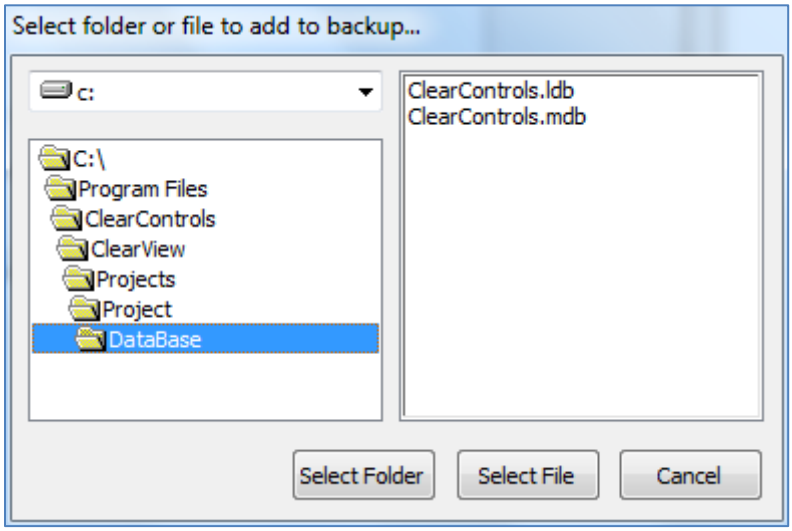


Figure #7.2

Remove

Select a file and click Remove when you want to delete a file from the list of files to be backed up.

Option

The Option tab lets users select where the files will be copied to when they are backed up. Any folder on the computer can be used to hold files.

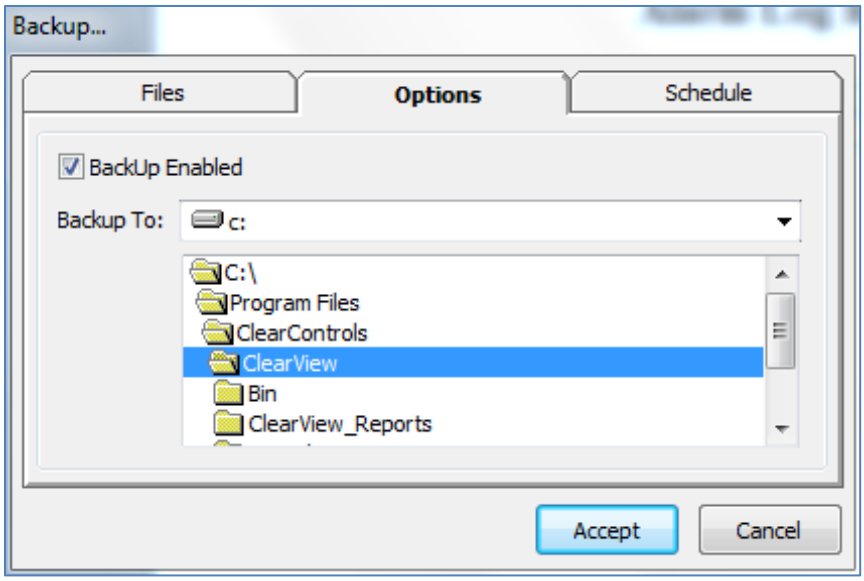


Figure #7.3

Backup Enabled

The check box Backup Enabled appears dimmed if no files are in the file list in the Files tab. When the user selects a file the Backup Enabled check-box is usable and is used to control whether the files are saved or stored. This option is convenient for users who have many files to back up. They can make the large list and then use the check box to control if the backup happens instead of having to create a large list every time they want to backup files.

Schedule

The Schedule tab lets a user set a time and interval for the backup or set a tag-based event to trigger the backup.

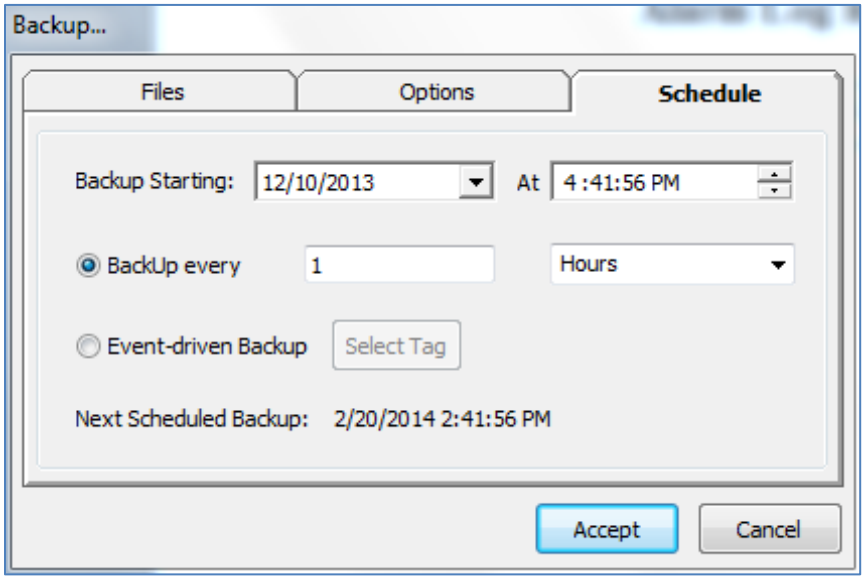


Figure #7.4

Backup Starting

The Backup Starting area has a drop-down menu you can use to select a date and time for the beginning of a backup.

Backup every

The Backup every area has a number field and drop-down menu to schedule backups on a time interval.

Event-driven Backup

The Event-driven Backup has a Select Tag button and a display box to show what tag was selected.

The Select Tag window lets you select from a list of tags and has a filter if you only want to view a smaller number of tags.

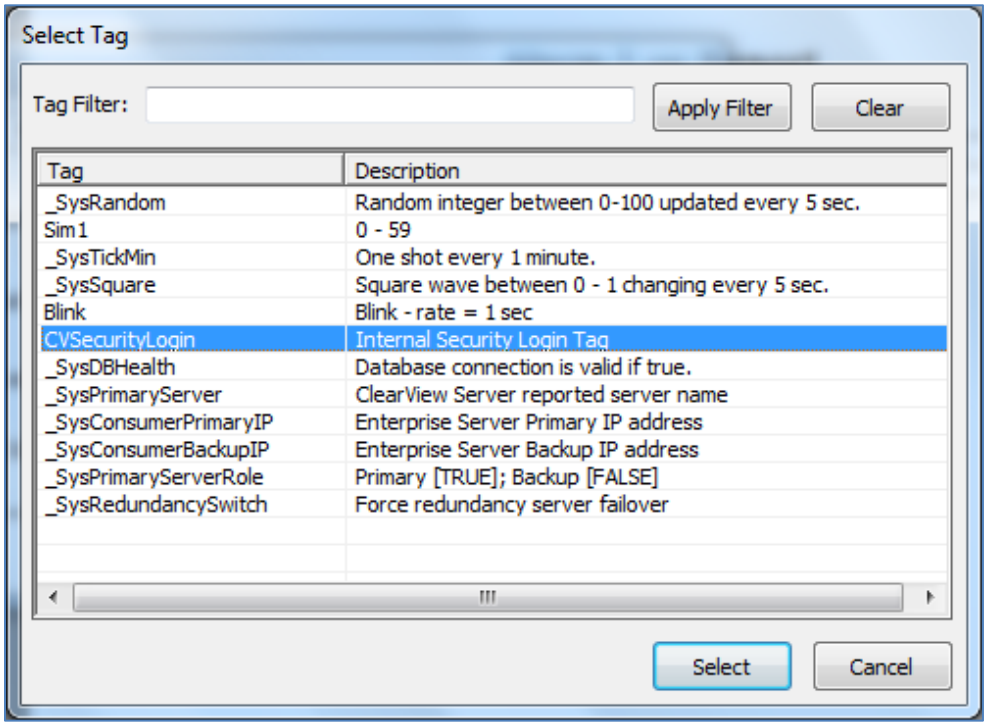


Figure #7.5

SECTION 8

Project Backup/Restore

In order to move the project to a different location/host it is recommended to use default ClearView-SCADA Project Backup and Restore functions

Project Backup

To create a project backup

1. Create a folder somewhere on your system.
2. Open the project which you have plans to backup.
3. On the Main Menu select Backup Project.

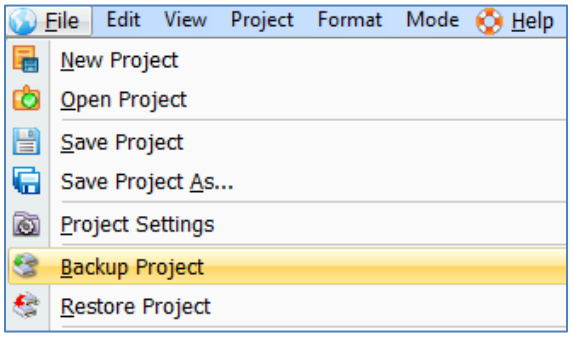


Figure #8.1

4. Specify the folder where the backup file would be stored – Figure #8.2.
5. If it is required additional files can be included into Backup Project by clicking on "Files…" button – Figure 8.3. Such files could be for example GIS Configuration maps used in the project, reports and any other files related to a project.

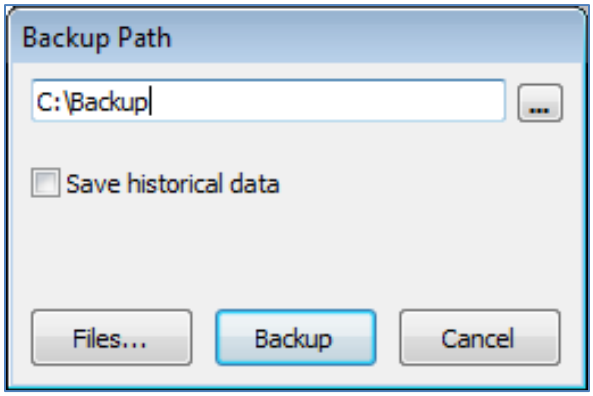


Figure #8.2

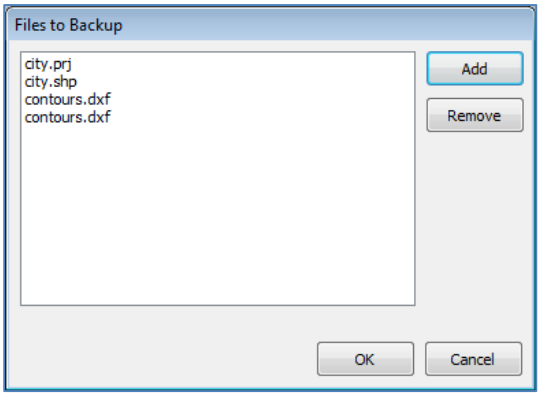


Figure #8.3

6. After clicking on "Backup" button the .zip files will be created which later can be used to restore the project.

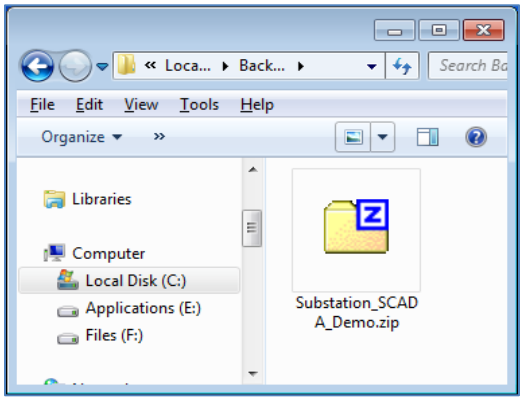


Figure #8.4

Project Restore

To restore the project:

1. Create a folder somewhere on your system where you have plans to restore the project.
2. Through Main Menu select Restore Project.

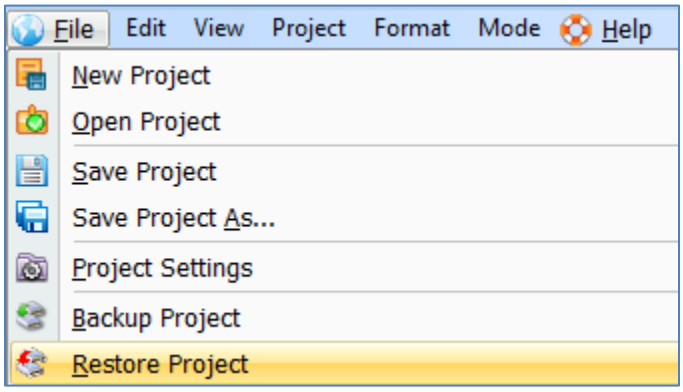


Figure #8.5

Restoring Project to Microsoft Access Database

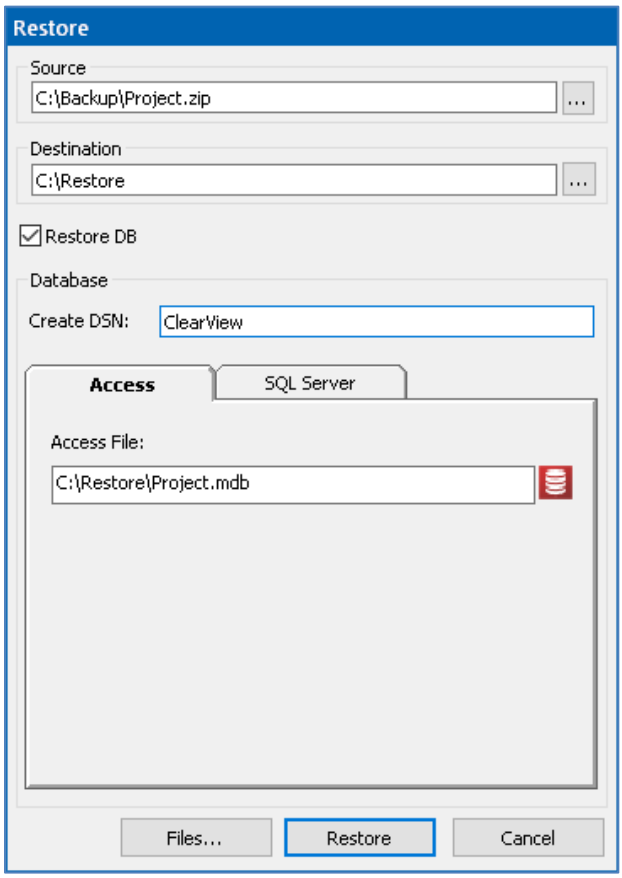


Figure #8.6

1. Create a directory where the project would be restore to
2. Specify the “Source” file (location of your backup)
3. Specify “Destination” where the project would be restored to
4. Specify a new DSN
5. If some previously backup files are not required ClearView-SCADA provides a feature to deselect them by clicking on “Files…” button

Restoring Project to Microsoft SQL Server Database

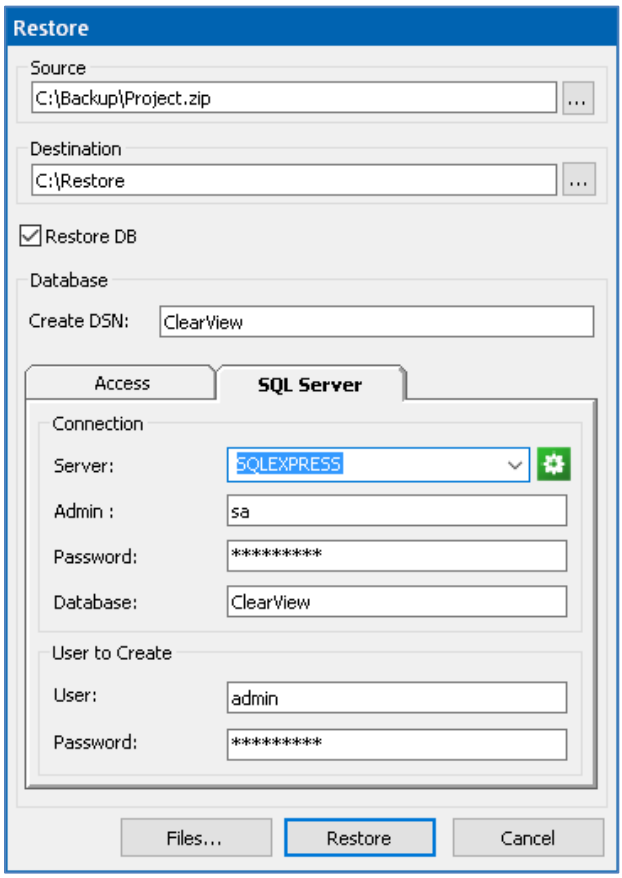


Figure #8.7

1. Create a directory where the project would be restored to
2. Specify the “Source” file (location of your backup)
3. Specify “Destination” where the project would be restored to
4. Specify a new DSN
5. Specify SQL Server Name
6. Specify SQL Server Administrative User Name and Password
7. Enter a new name for the SQL Server database
8. Optional – create additional user with privileges only to created database by specifying User Name and Password
9. If some previously backup files are not required ClearView-SCADA provides a feature to deselect them by clicking on “Files…” button

SECTION 9

Designing a Graphical Interface

Screens

Screens are windows containing controls. For example, controls may be graphical objects such as a pipe, report controls, or trend controls. After a project has been created, users with security privileges may add or edit screens. After a screen is created, objects shown in the Object Library, the Dynamo Library, or the Shapes Toolbar can be placed on the screen's Drawing Pad.

Each individual screen can be placed in Design or Run mode. When a saved project is opened, all screens are placed in Run mode.

Screen Settings

Screen settings may be set while in Design or Run mode, or programmatically through scripting.

Add a Screen

1. On the **Project** menu, click **Add Screen**.
2. Press **OK** when done with screen settings

Saving Screens

Screens are saved when the project is saved.

Opening/Showing a Screen

1. Open Application Explorer.
2. Double-click the screen you want to open or Right-click the screen you want to rename, and then click Open.

Note: A screen can also be opened programmatically using the script:

Screens.OpenScreen *screen-name*.

Rename a Screen

1. Open **Application Explorer**.
2. Right-click the screen you want to rename, and then click **Edit**. The Edit Screen dialog box appears.
3. Change the screen name to a unique name and press **OK**.

Delete a Screen

1. Open **Application Explorer**.
2. Right-click the screen you want to delete, and then click **Delete**.

Export a Screen

1. Open **Application Explorer**.
2. Right-click the screen you want to export.

3. On the File menu, select **Export**.
4. Enter a filename and press **Save**.

Import a Screen

1. Open **Application Explorer**.
2. On the File menu, select **Import**. The Open File dialog box appears.
3. Select the file to import and press **Open**.
4. Provide a unique screen name and press **OK**.

Change Screen Size, Position, and Behavior

1. Open **Application Explorer**.
2. Right-click the screen name you want to change.
3. Click **Edit**. The Edit Screen appears (Figure #9.1).

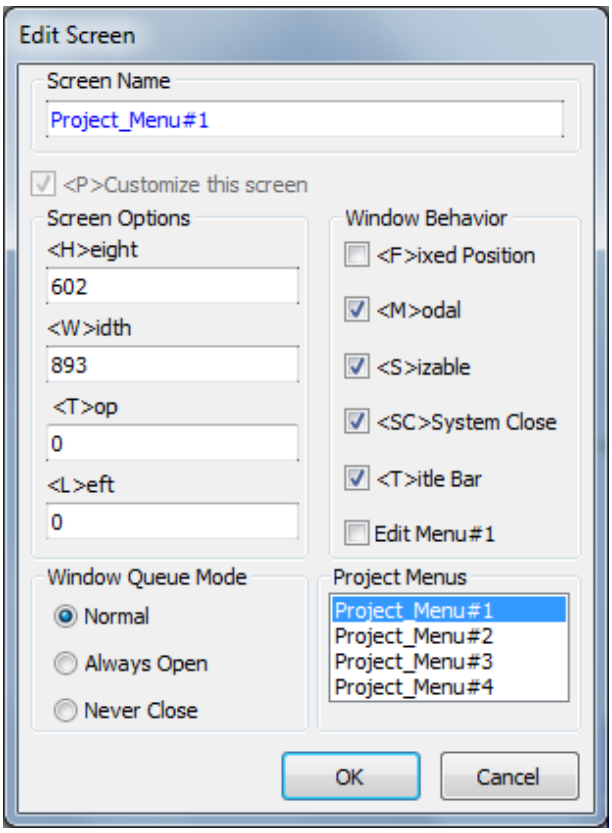


Figure #9.1

4. On the Edit Screen dialog box, check **Customize this Screen**.
5. Set the widow Height, Width, Top, and/or Left positions by manually entering pixel values.
6. Check the following window behavior properties:
   - **Fixed Position** – Determine if the screen is movable.
   - **Modal** – Window always has focus within ClearView

- **Sizeable** – Screen size is changeable while in Run mode.
- **System Close** – Disable the "X" (close button) on the title bar
- **Title Bar** – Display the title bar with screen name and close button.
- **Edit Menu** – Allows User to place objects on "Menu" screens.
- **Project Menus** – Project menus selection (always on top).

Note: Saving a project saves the last screen size. Use Scripting to maintain desired size.

7. Press **OK** when done with window screen settings.

Tip: You can also size a screen by enabling **Sizable** and disabling Fixed Position; then in Run mode, resize the screen to the desired size and position by dragging its borders. Finally, disable **Sizable** and set screen to **Fixed Position**.

Tip: Four popup screens are reserved for project menus (always on top)

- Project_Menu#1
- Project_Menu#2
- Project_Menu#3
- Project_Menu#4

To switch Project Menu screens in Design mode, check **Edit Menu**. To switch back to Run mode, uncheck **Edit Menu**.

Note: A project saved after a screen is resized (opens the screen at the new position and size).

Resizing an Open Screen

- To change the width, point to the left or right screen border. When the pointer changes into a horizontal double-headed arrow, drag the border to the right or left.
- To change the height, point to the top or bottom screen border. When the pointer changes into a vertical double-headed arrow, drag the border up or down.
- To change the height and width simultaneously, point to any screen corner. When the pointer changes into a diagonal double-headed arrow, drag the border in any direction.

Note: You cannot resize a screen when its window behavior property is not set to Sizable or Fixed. A project saved after a screen has been resized will open the screen at the new position and size.

Window Queue Mode

Each screen has a **Window Queue** mode, settable from the Create or Edit Screen dialog box. These settings only apply if the **Screen Queue Size** (project setting) is greater than 0.

To change the Window Queue mode:

1. Open **Application Explorer**.
2. Right-click the screen you want to change, then click **Edit**. The Edit Screen dialog box appears.
3. Check one of the following modes:

- **Normal** – The screen will be loaded and unloaded into memory as dictated by the Screen Queue Size (project setting).
- **Always Open** – The screen will be loaded into memory when the project is opened, and will always be loaded into memory.
- **Never Close** – The screen will not be loaded into memory until it is opened for the first time. Once opened it will always remain in memory.

1. Press **OK** when done with window screen settings

Drawing Pad

Each screen consists of a Drawing Pad available when in Design mode. Color, background picture, design grid, script update, and debug mode properties are available at that time.

Hide/Display Properties Window

1. Open a screen.
2. Switch screen to **Design** mode.
3. On the View menu, click **Properties** to hide or display the properties window.

Viewing Group Properties

1. Select a group of objects.
2. Right-click one of the objects in the group, then right-click and select Properties from the drop-down menu to display the Common Properties for the selected object.

Change Screen Color

1. Change screen to **Design** mode.
2. Right-click the Drawing Pad space, then click **ForeColor** or **BackColor** in the Properties window.

   Note: ForeColor is the Grid color.

3. Select new color.

Add a Background Picture

1. Change screen to **Design** mode.
2. Right-click the Drawing Pad space, then click the **Picture** property in the Properties window.
3. Select the picture file to use for the background picture, then press **Open**.

   Note: The **Use Picture** property can be used to hide the picture. This does not, however, remove the picture from the project's memory.

Remove Background Picture

1. Change screen to **Design** mode.
2. Right-click the Drawing Pad space, then click the **Picture** property in the **Properties** window.
3. Select an icon (*.ico) type file, then press **Open**. The picture has been removed from the pad's memory.

Changing Grid Properties

The design Grid is available in Design mode.

1. Change to **Design** mode.
2. Right-click the Drawing Pad space to update the properties window with the pad properties. Then click the Grid property in the Properties window.
3. Or use the Mode Menu to set Grid properties.

Changing Script Update Interval

The Drawing Pad property, **ScriptUpdateInterval,** controls the time interval between update events.

1. Change the screen to **Design** mode.
2. Right-click the Drawing Pad space, then click the **Script Update** property in the **Properties** window.
3. Enter the quantity of milliseconds for updating the scripts. Default: 500 milliseconds.

Changing Debug Mode

Turning on the Debug mode will enable troubleshooting the scripted code contained in the current window.

1. Change the screen to **Design** mode
2. Right-click the Drawing Pad space, then set the **Debug** mode property in the Properties window to **True** or **False**
3. Return the screen to **Run** mode.

Drawing Tools

The following drawing tools are available (in order of appearance on Figure #9.2): line, arch, circle, triangle, rectangle, rounded rectangle, free curve, free shape, and custom multisided shapes (the name "**Customize**" is used at the time of mouse pointing to an icon). Each tool has settable properties. Once drawn, a drawing tool can be reshaped, resized, or rotated.

Figure #9.2

	Line
	Arch
	Circle
	Triangle
	Rectangle

	Rounded Rectangle
	Free curve
	Free shape
	Custom multisided shapes

Changing Drawing Tool Properties

Changes take effect for the Drawing Tools drawn after the default settings are set.

1. Click **Customize** icon**.**
2. The **Shape Default** dialog box appears (Figure #9.3).

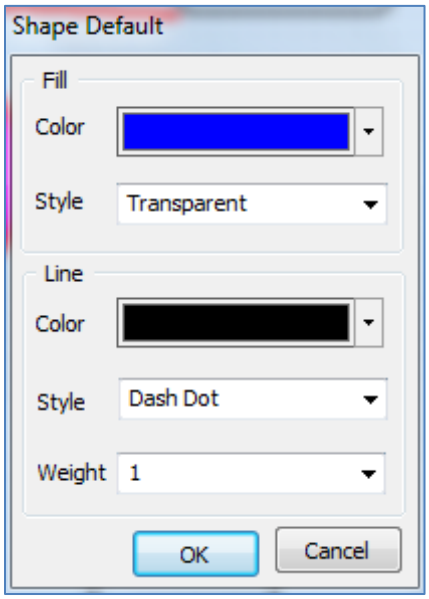


Figure #9.3

3. Press **OK** after changing the default settings
   - **Fill Color** – Sets closed objects fill color
   - **Fill Style** - Select between Opaque and Transparent
   - **Line Color** – Sets the line color
   - **Line Style** – Sets the line style (Solid, Dash, Dot, Dash Dot, Dash Dot Dot)
   - **Weight** – Sets pixel width of the Solid border only; for the rest of the Line Styles it will overwrite Line Style type to Solid if Weight is not equal to 1.

Resize, Reshape, or Rotate a Drawing Tool

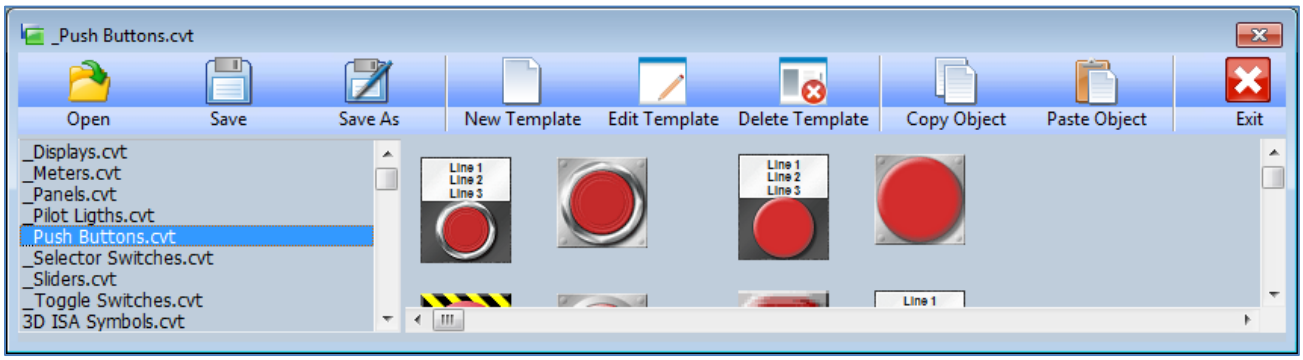
1. In **Design** mode, select the drawn tool.

2. Right-click the Drawing Tool, select **Design Frame**, and then click "**Reshape**", "**Resize**" or "**Rotate**" Frame. The object will now have a frame with pick points.
3. Use the mouse to grab a pick point and drag it to reshape, resize or rotate the object.

Change Drawing Tool Properties

Once a Drawing Tool has already been placed on a screen's Drawing Pad, its properties are only changeable through the object's **Properties** window or via scripting. If additional tools are to be drawn with a different set of properties, then change the default settings.

Dynamos



The **Dynamo Library** contains hundreds of **ActiveX** objects licensed from **Software Toolbox Inc**. to be used with **ClearView**. The Dynamo library interface provides a menu list of ClearView template files (*.cvt), located in the /ClearView/Bin/Objects/CVT directory. These template files contain similar grouped objects. Custom template files can also be created and saved for access through the Dynamo Library interface.

Using a Dynamo Object

1. In **Design** mode, click **Dynamo** (see Figure #1.3). The Dynamo Object Library interface appears.
2. Click through any of the object library names listed to show the objects available.
3. Right-click an object, then select **Copy**, or use keyboard short cut, **Ctrl+C.**
4. To paste the object on the ClearView Drawing Pad, simply click the pad.

Note: Each dynamo has its own unique properties. To see a dynamo's properties, first right-click **Dynamo** then click an object to see its unique properties, if available.

Adding to the Dynamo Library

The Dynamo Library can be edited.

WARNING: If an original object is deleted from the library and the library saved, the object can only be restored by reinstalling ClearView.

To create a new Object template file (*.cvt):

1. In **Design** mode, click **Dynamo**. The Dynamo Object Library interface appears.
2. On Dynamo Object **File** menu, select **New**. A blank template opens.
3. Copy and paste an object or groups of objects from ClearView Drawing Pad to the template space.

4. On Dynamo Object **File** menu, select **Save As**.
5. Enter a filename and click **OK**.

To modify an existing Object template file (*.cvt):

1. In **Design** mode, click **Dynamo.**
2. Select from the list box the ClearView object template file to modify.
3. Modify (add, delete, rearrange) the object or groups of objects in the object template drawing space.
4. Once edits are complete, select **Save** on the Dynamo Object **File** menu or select **Save As**.
5. Enter a new filename and click **OK**.

WARNING: If an original object template file is deleted or saved over, the file can only be restored by reinstalling ClearView.

Object Library

ClearView provides a library of **ActiveX** objects. These objects are specifically designed for use with ClearView. ClearView **Object Library** can also be customized by deleting or adding ActiveX controls using the Object Toolbar Editor. The Object Toolbar Editor is available via the Application Explorer.

Open ClearView Object Library Editor

1. Open **Application Explorer**.
2. Expand the **Setup** folder and double-click **Object ToolBar Editor**. ClearView Object ToolBar Editor window opens (Figure #9.4).

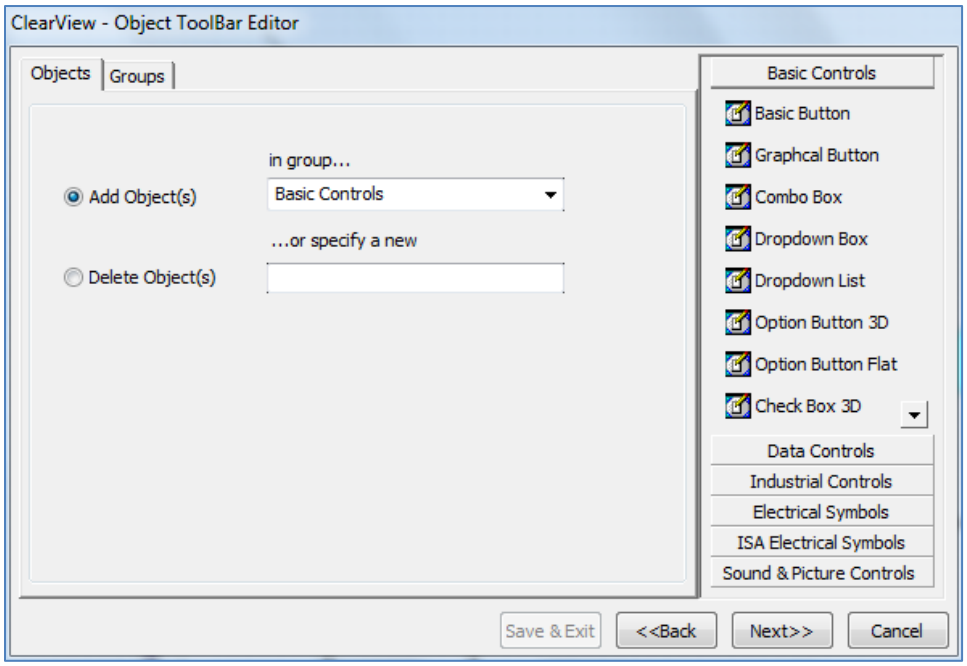


Figure #9.4

Adding an ActiveX control to Object Library

Controls are organized by placing them in groups.

1. Open the **Object ToolBar Editor** (Figure #9.4).
2. Select **Add Object(s**).
3. Specify a new group name or select the group name from the drop-down list box.
4. Click **Next**.

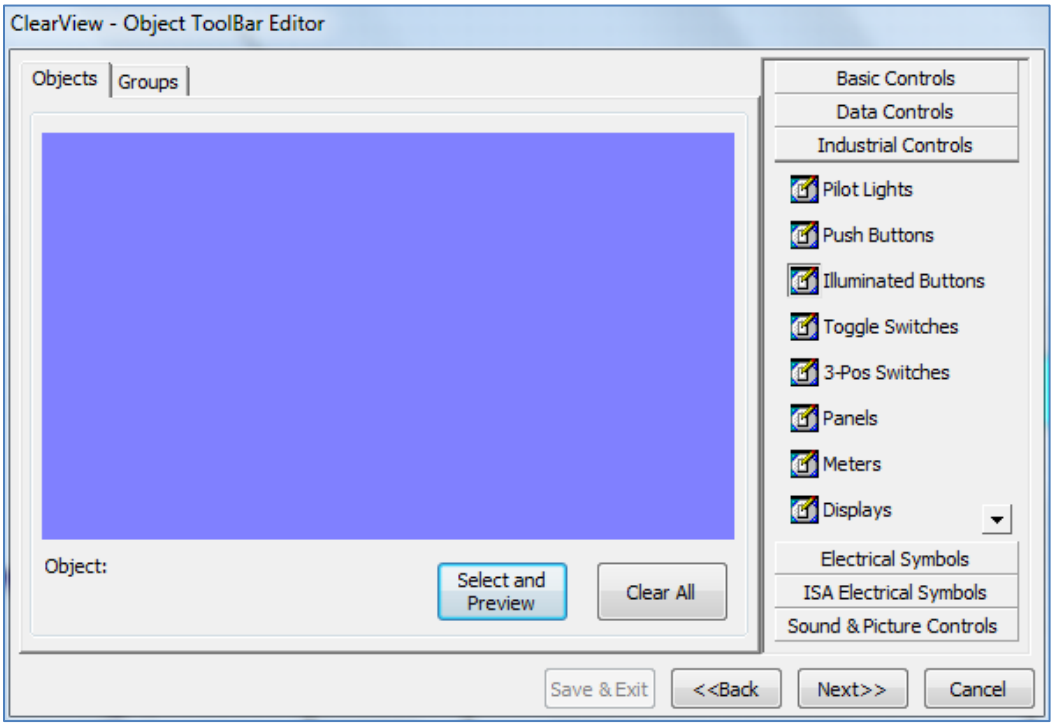


Figure #9.5

5. Click **Select and Preview** (Figure #9.5).
6. Locate the ActiveX file containing the control. Press **Open**. The object(s) appear(s) in the sample Drawing Pad.
7. Click **Next**.
8. Edit the **Caption** (control name) and **Tooltip Text** for each control, or use the default name.
9. Press **Accept**.
10. Press **Save & Exit** when done editing the Object Toolbar.

Tip: You can also select the group name by clicking the group shown in the **Object Toolbar Preview** window.

Note: If the file you selected has multiple ActiveX objects, then they will overlap each other. The object(s) can be moved around in the sample Drawing Pad. ClearView uses Windows regsvr32.exe to register the control.

Removing an ActiveX control from the Object Library

1. Open the **Object ToolBar Editor**.
2. Select **Delete Object(s)** (see Figure #9.4).
3. Select the group name and then the object name to be deleted from the dropdown list boxes
4. Click **Next**.

5. Press **Accept**.
6. Press **Save & Exit** when done editing the Object Toolbar.

Tip: You can also select the object or group by clicking the group and object shown in the **Object Toolbar Preview** window.

Keyboard Shortcuts

Shortcut	Description
[Ctrl + X]	Cut the selected object(s)
[Ctrl + C]	Copy the selected object(s)
[Ctrl + V]	Paste object
[Delete]	Delete the selected object(s)
[Ctrl + Z]	Undo the last action
[Ctrl + Y]	Redo the last action
[Ctrl + F]	Bring object to front
[Ctrl + B]	Send object to back
[Shift + F]	Bring object forward (one layer)
[Shift + B]	Send object backward (one layer)
[Shift + R]	Snap object(s) to Grid
[Ctrl + R]	Align object(s) to Grid
[Ctrl + →]	Move the selected object “Right”
[Ctrl + ←]	Move the selected object “Left”
[Ctrl + ↑]	Move the selected object “Up”
[Ctrl + ↓]	Move the selected object “Down”
[Shift + ↑]	Resize the selected object Width “Larger”
[Shift + ↓]	Resize the selected object Width “Smaller”
[Shift + →]	Resize the selected object Height “Larger”
[Shift + ←]	Resize the selected object Height “Smaller”

Shortcut	Description
[Shift + Mouse Object Resize]	Resize the selected object “Proportionally”
[Ctrl + S]	Select All objects on the screen

SECTION 10

Scripting

ClearScript was created to extend the capabilities of ClearView SCADA. Scripting increases flexibility with useful functionality. ClearScript scripting uses VB Script, a high-level scripting language developed by Microsoft®.

There are several ClearScript properties available. To access them, make sure you are in Design mode and there is no screen objects selected. The properties box will then list the following properties: *ScriptUpdate, ScriptUpdateInterval*, and *DebugMode*.

ScriptUpdate returns or sets a Boolean value indicating whether the CCPad is sending an Update event.

*ScriptUpdateInterva*l returns or sets the interval between subsequent Update events.

DebugMode indicates if the script debugger will start on script execution. If DebugMode = False, then all syntax and run-time errors will be bypassed (the script would not execute). To bypass run-time errors, use **"On Error Resume Next"** in your script. This procedure will bypass errors such as division by zero, overflow, wrong data type, etc.

In the ClearScript Editor, Procedure mode displays code only for a particular procedure and Full mode displays all code. There are two ways to enter ClearScript:

You can right-click an object and select **ClearScript** on the menu. This will start ClearScript in Procedure mode and only the code for the selected object.

You can unselect all objects, right-click the workspace, and choose **ClearScript** from the menu. This action will start ClearScript in Full mode and all code will be displayed

Note: Refer to MSDN article on VBScript Users Guide and VBScript Language Reference.

ClearScript Editor

To view the ClearScript Editor (Figure #10.1) double-click the Drawing Pad work area, or right-click the Drawing Pad work area and select **Edit Script**.

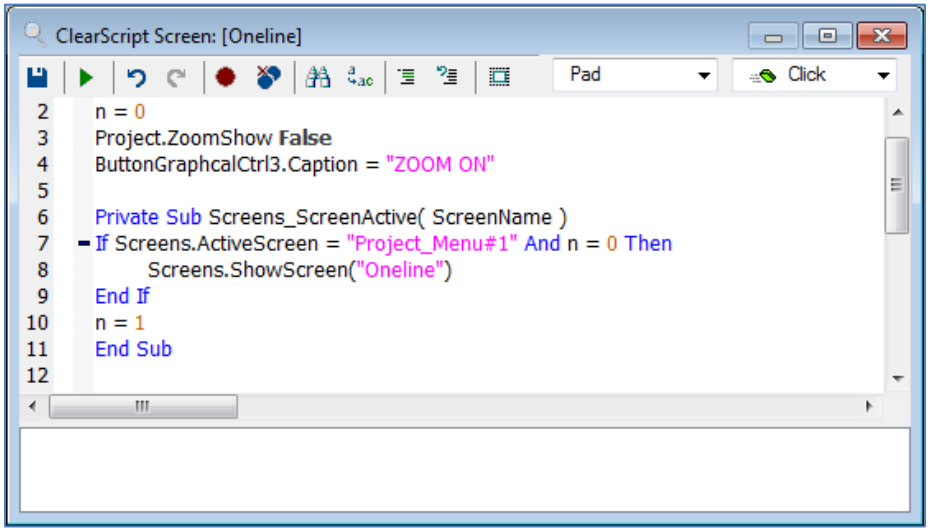


Figure #10.1

ClearScript Editor Toolbar

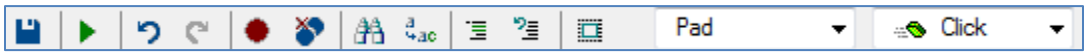


Figure #10.2

	Save (available when changes are done to the script).
	Run/Debug Script
	Undo Changes
	Redo Changes
	Insert/Remove Breakpoint (F9)
	Remove All Breakpoints
	Find Text
	Replace text
	Comment out selected lines
	Uncomment out selected lines
	Insert External COM Objects

ClearScript Editor Popup Menu

Right-click in the ClearScript editor space to view additional options.

Objects	Opens a dialog box which displays a list of COM Objects which may be added to the scripting
Cut	Cut script text
Copy	Copy script text
Paste	Paste script text
Print	Print the script text
Undo	Object View – toggles Procedure/Full Mode View
Redo	Find – search text within the code
Find...	Replace – search and replace text within the code
Replace...	
Insert/Remove Breakpoint (F9)	

	Insert/Remove Breakpoint

ClearScript Debug Mode

Click on Drawing Pad area when it is set to Design mode. In Properties for DrawingPad set option DebugMode to True. Change back to Run mode. If script has an error, it'll open Clear Script Debugger window (Figure #10.3)

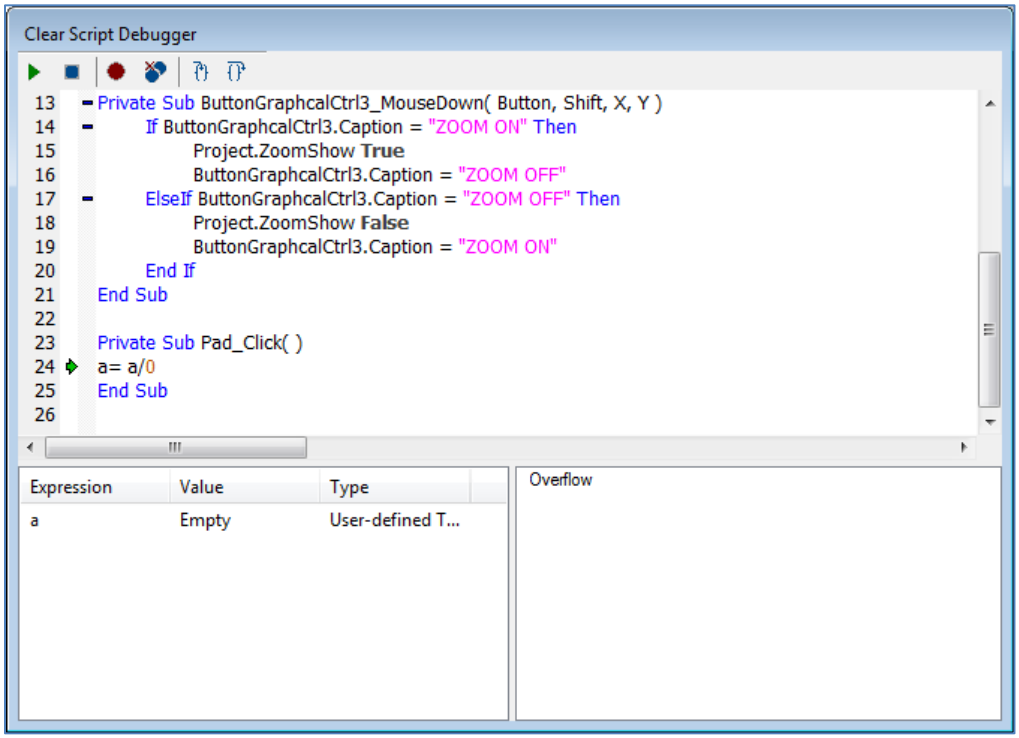


Figure #10.3

Internal Variable Window

Clear Script Debugger window has additional window - easily view variable values and types in this window. It is located at the bottom of the debugger window (Figure #10.3).

ClearView Intrinsic Objects

ClearView objects allow script users to interact with ClearView by viewing forms, settings tags, and alarms and by changing security settings. All object properties are accessible in scripting along with their events and methods.

Drawing Pad Object (Workspace)

Properties

Description	BackColor property gets or sets the background color of the Pad Object
Syntax	Pad.BackColor [= color]
Data Type	OLE_COLOR
Environment	Run/Design Time
Access	Read/Write

Description	UsePicture property gets or sets the background picture visibility
Syntax	Pad.UsePicture [= True]
Data Type	Boolean
Environment	Run/Design Time
Access	Read/Write

Description	Picture property gets or sets the background picture visibility
Syntax	Pad.Picture [= LoadPicture(“picture_name”)]
Data Type	Picture object
Environment	Run/Design Time
Access	Read/Write

Description	ScriptUpdate property gets or sets execution of the script (CCPadUpdate)
Syntax	Pad.ScriptUpdate = TRUE/FALSE
Data Type	Boolean
Environment	Run/Design Time
Access	Read/Write

Description	ScriptUpdateInterval property gets or sets script update interval in milliseconds (CCPadUpdate)
Syntax	Pad.ScriptUpdate = 500
Data Type	Long
Environment	Run/Design Time
Access	Read/Write

Description	PanEn property gets or sets pan enabling used for partial ZOOM
Syntax	Pad. PanEn = TRUE/FALSE
Data Type	Boolean
Environment	Run/Design Time
Access	Read/Write

Description	PanColor property gets or sets pan frame color
Syntax	Pad. PanColor = 255
Data Type	OLE Color
Environment	Run/Design Time
Access	Read/Write

Description	EnableZoom property gets or sets ZOOM functionality
Syntax	Pad. EnableZoom = TRUE/FALSE
Data Type	Boolean
Environment	Run/Design Time
Access	Read/Write

Description	FitMode property gets or sets Fit mode functionality
Syntax	Pad. FitMode = 0 ‘Original Size

	Pad. FitMode = 1 'Fit Screen Pad. FitMode = 2 'Fit Width Pad. FitMode = 3 'Fit Height Pad. FitMode = 4 'Zoom the entire screen
Data Type	Integer
Environment	Run/Design Time
Access	Read/Write

Description	Zoom property gets or sets ZOOM magnification
Syntax	Pad. Zoom = 200
Data Type	Long
Environment	Run/Design Time
Access	Read/Write

Methods

Description	Pad.Controls.Item(i) method addresses Intrinsic properties of the Pad's item

Syntax	For i = 0 To Pad.Controls.count
	  If Pad.Controls.item(i).Name = "BRK18" Then
	  Pad.Controls.item(i).Object.BackColor = **vbRed**
	  End If
	Next

Parameter(s)	N/A

Description	Pad.Controls.Item(i).Object method addresses custom properties of the Pad's item

Syntax	For i = 0 To Pad.Controls.count
	  If Pad.Controls.item(i).Name = "BRK18" Then
	  Pad.Controls.item(i).Object.BackColor = **vbRed**
	  End If
	Next

Parameter(s)	N/A

Events

Description	CCPadUpdate event occurs based on preset ScriptUpdateInterval Note: CCPadUpdate Applies to all Pad Object, ActiveX object and ActiveX Groups
Syntax	Sub Pad_CCPadUpdate( ) 'Your Code here End Sub
Parameters	None
Environment	Run-Time Only

Note: All standard events are implemented: Key Events, Click Events, and Mouse Events

Application Object

Properties

None.

Methods

Description	ExitClearView method instructs ClearView to exit
Syntax	'Application.ExitClearView Private Sub Object1_Click( ) Application.ExitClearView End Sub Note: If the project has changed, the save Project dialog box will appear before shutting down.
Parameter(s)	N/A

Description	OnScreenKeyboard method opens on-screen keyboard
Syntax	‘Application. OnScreenKeyboard Private Sub Object1_Click( ) Application. OnScreenKeyboard End Sub
Parameter(s)	N/A

Description	OpenProject (Path) method opens a project from a location path
Syntax	‘Application.OpenProject Private Sub Object1_Click( ) Application.OpenProject “C:\Project\MyProject.cvp” End Sub Note: If the currently opened project has changed, the save Project dialog box will appear before loading the new one.
Parameter(s)	Path – String: path to the project file

Description	GetAvailPhysRAM method returns available physical RAM
Syntax	Application.GetAvailPhysRAM
Parameter(s)	N/A

Description	GetTotalPhysRAM method returns total RAM
Syntax	Application.GetTotalPhysRAM
Parameter(s)	N/A

Description	GetUsedPercentRAM method returns used RAM (%)
Syntax	Application.GetUsedPercentRAM
Parameter(s)	N/A

Description	GetLocalComputerName method returns local computer name
Syntax	Application.GetLocalComputerName
Parameter(s)	N/A

Description	GetLocalIP method returns local computer IP address
Syntax	Application.GetLocalIP
Parameter(s)	N/A

Description	GetTotalCPU_Load method returns total CPU load
Syntax	Application.GetTotalCPU_Load
Parameter(s)	N/A

Description	ShellApp (Path) method opens/runs any executable file (with parameters)
Syntax	‘Application.ShellApp Private Sub Object1_Click( ) Application.ShellApp “C:\Project\MyProject.exe /n” End Sub
Parameter(s)	Path – String: path to the executable file

Events

None.

Screens Object

Properties

Description	ActiveScreen property gets the Active Screen Name

Syntax	‘Screens.ActiveScreen
	Sub Pad_CCPadUpdate( )
	  If Screens.ActiveScreen = “Screen Name” Then
	  ‘Your Code Here
	  End If
	End Sub

Data Type	String
Environment	Run Time
Access	Read Only

Description	AuditScreens property enables or disables logging screen changes by a user
Syntax	‘Audit screen is enabled Screens. AuditScreens = True
Data Type	Boolean: True – enable screens change logging, False – disable screens change logging
Environment	Run Time
Access	Read/Write

Methods

Description	ShowScreen method opens ClearView screen
Syntax	‘Screens.ShowScreen(“Your Screen Name”) Private Sub Object1_Click( ) Screens.ShowScreen(“Your Screen Name”) End Sub
Parameter(s)	ScreenName - String: Screen Name

Description	CloseScreen method closes ClearView screen
Syntax	‘Screens.CloseScreen(“Your Screen Name”) Private Sub Object1_Click( ) Screens.CloseScreen(“Your Screen Name”) End Sub
Parameter(s)	ScreenName – String: Screen Name

Description	ScreenLeft(m_ScreenName) method returns specified screen left position of a Long type
Syntax	X = Screens.ScreenLeft(“Screen_Name”)
Parameter(s)	m_ScreenName – String: Screen Name

Description	ScreenTop(m_ScreenName) method returns specified screen top position
Syntax	X = Screens.ScreenTop(“Screen_Name”)
Parameter(s)	m_ScreenName - String: Screen Name

Description	ScreenWidth(m_ScreenName As String) method returns specified screen width
Syntax	X = Screens.ScreenWidth(“Screen_Name”)
Parameter(s)	m_ScreenName - string

Description	ScreenHeight(m_ScreenName) method returns specified screen height
Syntax	X = Screens.ScreenHeight("Screen_Name")
Parameter(s)	m_ScreenName – String: Screen Name

Description	ScreenPosSize(m_ScreenName , Left, Top, Width, Height) method sets screen position and size
Syntax	Screens.ScreenPosSize "Screen_Name", Left, Top, Width, Height
Parameter(s)	Screen_Name – string, Left, Top, Width, Height - integer

Description	PublicMethod(n_GlobalVar1, n_GlobalVar2, GlobalVar3, n_GlobalVar4) is global to all screens. The user is capable of passing any information between ClearView screens including ActiveX object reference, arrays or any other data types. When the method is executed the PublicEvent event will be raised notifying all ClearView screens that the data is changed. Note: It's not necessary to pass any information to the other screens – Optional Warning: When addressing ActiveX controls via PublicMethod make sure that the referenced control is available. Make sure that the object exists (not deleted) and the screen is not unloaded (check the screen properties)
Syntax	Screens.PublicMethod ScreeName_Array, Integer_Array, Object_Array, "String"
Parameter(s)	Reference

Description	Screens.ImportForm ("C:\temp\trend1.ccs", 1, 1, 1, 1,0,1,100,100,100,100). The user is capable of importing any previously exported screen (.ccs) programmatically via. Scripting. Notes: • If Parameter 2 (prop_Popup) = 0 or not specified parameters 3 to 11 are disabled and function would create a full screen • If Parameter 2 to 7 is numeric (0 or 1) values 0 – Disabled; 1 - Enabled • If Parameter 8 to 1 is numeric unsigned values • If any of the parameters 8-11 (size/position) and Parameter 2 (prop_Popup) = 1 the function would create a default customizable screen with default parameters • Parameters 8 – 11 represented in pixels • Function creates and add to screen collection the form named "User-Forms" • Function returns TRUE if the new screen was created successfully or False if error occurred during creation
Syntax	Private Sub Button_Click( ) Dim tmp tmp = Screens.ImportForm ("C:\temp\trend1.ccs", 1, 1, 1, 1,0,1,100,100,100,100) If tmp Then Screens.ShowScreen "User-Forms" End Sub
Parameter(s)	See below

No.	Parameters	Description	Data Type	Required	Value
1	FilePath	Specifies location of .ccs file	String	YES	Path
2	prop_Popup	Specifies if the screen would be customizable	Integer	Optional	0/1
3	prop_Modal	Specifies customizable screen would be modal	Integer	Optional	0/1
4	prop_Sizable	Specifies customizable screen would sizable	Integer	Optional	0/1
5	prop_SystemClose	Specifies customizable screen would System Close	Integer	Optional	0/1
6	prop_FixedPos	Specifies customizable screen would display at fixed position	Integer	Optional	0/1
7	prop_Titlebar	Specifies customizable screen would display Title Bar	Integer	Optional	0/1
8	prop_Height	Specifies the Height of the customizable screen	Integer	Optional	Numeric
9	prop_Width	Specifies the Width of the customizable screen	Integer	Optional	Number

No.	Parameters	Description	Data Type	Required	Value
10	prop_Left	Specifies Left position of the customizable screen	Integer	Optional	Number
11	prop_Top	Specifies Top position of the customizable screen	Integer	Optional	Number

Events

Description	ScreenActive event occurs when the screen is activated

Syntax	‘Private Sub Screens_ScreenActive(ScreenName)
	
	Private Sub Screens.ScreenActive(ScreenName)
	  If ScreenName = “Your Screen Name” Then
	  ‘Your Code Here
	  End If
	End Sub

Parameter(s)	ScreenName – Returns Screen Name

Description	ScreenDeActive event occurs when the screen is deactivated

Syntax	‘Private Sub Screens_ScreenDeActive(ScreenName)
	Private Sub Screens_ScreenDeActive(ScreenName)
	  If ScreenName = “Your Screen Name” Then
	  ‘Your Code Here
	  End If
	End Sub

Parameter(s)	ScreenName – Returns Screen Name

Description	PublicEvent(n_GlobalVar1, n_GlobalVar2, n_GlobalVar3, n_GlobalVar4) event is fired each time the PublicMethod is executed

Syntax	Private Sub Screens_PublicEvent(GlobalVar1, GlobalVar2, GlobalVar3, GlobalVar4)
	For I = 0 To UBound(GlobalVar1)
	If GlobalVar1(I) = "My Screen" Then
	GlobalVar2(0).Caption = "Your Text"
	End If
	Next
	End Sub

Parameter(s)	Reference

Project Object

Properties

Description	ClientGlobalAlarmState property gets current global alarm state as integer: 0 = No alarms 1 = All Alarms are Cleared and Not Acknowledged 2 = All Alarms are Active and Acknowledged 3 = New Active and Not Acknowledged Alarms in the system
Syntax	X = Project.ClientGlobalAlarmState
Environment	Run Time
Access	Read Only

Description	ServerConnTest property gets Boolean True if the client is connected to the primary or redundant server and False if it's disconnected.
Syntax	X = Project.ServerConnTest
Data Type	Boolean
Environment	Run Time
Access	Read Only

Description	PrimaryServerName property gets configured primary server name or IP address string value
Syntax	X = Project.PrimaryServerName
Data Type	String
Environment	Run Time
Access	Read Only

Description	RedundantServerName property gets configured redundant server name or IP address string value
Syntax	X = Project. RedundantServerName
Data Type	String
Environment	Run Time
Access	Read Only

Methods

Description	ConnectedServer(strServer) method returns **Boolean** connection state **True** or **False** based on specified server name (for example connected to primary or redundant server).
Syntax	X = Project.ConnectedServer("Server_Name")
Parameter(s)	strServer – String: Server Name

Description	DB_Reconnect method reconnects to the data base
Syntax	Project.DB_Reconnect
Parameter(s)	None

Description	WindowState(State) method sets the ClearView Client window state
Syntax	Project.WindowState 1
Parameter(s)	State – Integer: 0 = Normal, 1 = Minimized, 2 = Maximized

Description	DisableProjectSave(State) method disables or enables automatic project save routine
Syntax	Project.DisableProjectState True
Parameter(s)	State- Boolean: True = Disable, False = Enable

Description	IEExplorerNavigate(URL) method sets IE Explorer URL
Syntax	Project.IEExplorerNavigate "URL"
Parameter(s)	URL – String

Description	IEExplorerOpen(Open) method opens or closes Internet Explorer
Syntax	Project.IEExplorerOpen True
Parameter(s)	Open-Boolean: True – Opens IE, False – Closes IE

Description	IEExplorerPos(Left, Top , Width , Height) sets Internet Explorer position and size, return value - none
Syntax	Project.IEExplorerPos Left, Top, Width, Height
Parameter(s)	Left – Integer: left window side X coordinate in pixels Top – Integer: top window side Y coordinate in pixels Width – Integer: window width in pixels Height – Integer: window height in pixels

Description	ProjectDirectory method returns path to project directory as **String**
Syntax	Project.ProjectDirectory
Parameter(s)	None

Description	ProjectPath method returns path to project file as String
Syntax	Project.ProjectPath
Parameter(s)	None

Description	ProjectReconnect (DSN, Server [, UserName, UserPassword, DataProvider]) method reconnects to the specified project, return value- none
Syntax	Project.ProjectReconnect DSN, Server, UserName, UserPassword, DataProvider
Parameter(s)	DSN – String: Data Source Name Server – String: ClearView Server IP address UserName – String: optional data base user name UserPassword – String: optional data base user password DataProvider – String: optional data provider, should be mandatory and specified as "OODBC" for Oracle data base

Description	MillisecondSystemTime method returns system time in milliseconds
Syntax	mTime = Project.MillisecondSystemTime
Parameter(s)	None

Description	MillisecondTime (Time) method returns millisecond time as String
Syntax	sTime = Project.MillisecondTime (Time)
Parameter(s)	Time – time in milliseconds

Description	AddDomainUsers (Username, FirstName, LastName, UserGroup) method returns **Boolean True** if user is added and **False** if user is not added.
Syntax	nUsersAdded = Project. AddDomainUsers ("jdoe" ," John", "Doe", "Administrators")
Parameter(s)	UserName – String FirstName – String LastName- String UserGroup -String

Description	ImportDomainUsers (path) method imports users from .csv file and returns the number of successfully imported users as **Long**.
Syntax	nUsers = Project. ImportDomainUsers (“C:\documents\users.csv”)
Parameter(s)	path – String: path to the .csv file to import

Description	ShowMap (MapName) method shows configured GIS map
Syntax	Project.ShowMap (“MapName”)
Parameter(s)	MapName – String: configured GIS map name

Description	DisableAlarmGroup (GroupName, mDisable) method disables or enables alarm group, return value - none
Syntax	Project.DisableAlarmGroup “CVAlarm”, True
Parameter(s)	GroupName – String – alarm group name mDisable – Boolean: True- disable group, False – enable group

Description	DisableReconnectMsg (mDisable) method disables or enables reconnect message, return value - none
Syntax	Project.DisableReconnectMsg(True)
Parameter(s)	mDisable – Boolean: True- disable message, False – enable message

Description	MonitorPos (Monitor) sets monitor position for the project, return value - none
Syntax	Project.MonitorPos (1)
Parameter(s)	Monitor – Long: 1- Primary Monitor I, 2 – Secondary Monitor II

Description	MousePos (Monitor) sets mouse position either to primary or to the secondary monitor, return value - none
Syntax	Project.MousePos (1)
Parameter(s)	Monitor – Long: 1- Primary Monitor I, 2 – Secondary Monitor II

Description	ZoomLensShape (Shape) method sets the magnifying lens shape, return value - none
Syntax	‘Setting lens shape to square Project.ZoomLensShape(0)
Parameter(s)	Shape – Long: 0 - square shape, 1 – circle shape

Description	ZoomLensSize (Width, Height) method sets the magnifying lens size in pixels, return value - none
Syntax	‘Setting lens size to 200 x 300 Project.ZoomLensSize 200, 300
Parameter(s)	Width – Long: Lens width in pixels Height – Long: Lens height in pixels

Description	ZoomMagnification (Level) method sets the magnifying lens magnification level in percent
Syntax	‘Setting lens magnification to 200% Project.ZoomMagnification(200)
Parameter(s)	Level – Long: Magnification level

Description	ZoomShow(Show) method shows or hides magnifying lens
Syntax	‘Showing magnifying lens Project.ZoomShow(True)
Parameter(s)	Show– Boolean: True – show, False - hide

Description	ZoomViewMode(Mode) method sets the zoom view mode to either a lens or full screen
Syntax	‘Setting zoom view mode to lens Project.ZoomViewMode(Mode)
Parameter(s)	Mode– Long: 0 – lens, 1 – full screen

Description	SetGISMapPath (map As String, Path As String) set's new path for a specific GIS map file
Syntax	Private Sub Pad_Click( ) Project.SetGISMapPath "GIS_DEMO", "C:\Program Files\ClearControls\ClearView\Projects\Project\EL_Distribution_Project\GIS\Map.ttkgp" End Sub
Parameter(s)	Map – Name of specific map. Path – Path to specific map

Description	ReadArray (TagValue, cDim) method reads array of data received from OPC, returns values in an array
Syntax	'Reading tags from the "ReadTag" tag name x = Project.ReadArray (Tags.item("ReadTag").Value, 0) For i = 1 To UBound(x) 'Your code can be placed here Next
Parameter(s)	TagValue – Variant: values of the tag CDim – Long: 0 – one dimensional array, 1– two-dimensional array

Description	WriteArray (Array) method returns compiled string that can be written to a tag
Syntax	Writing array to a tag "WriteTag" Tags.item("WriteTag").Value = Project.WriteArray(arr)
Parameter(s)	Array – array of values

Description	DBConfigMismatch method returns TRUE if the data bases of the ClearView Server(s) and the ClearView Client (s) aren't identical (tags and alarms) and FALSE if the data bases are identical
Syntax	Project.DBConfigMismatch
Parameter(s)	None

Description	AlarmEnable (AlarmName, AlarmEnable) function enables or disables alarms (database transaction only) and returns TRUE if transaction is successful or FALSE if transaction is NOT successful
Syntax	Project.AlarmEnable m_return = Project.AlarmEnable (AlarmName, False)
Parameter(s)	AlarmName – Alarm name (string) AlarmEnable – Alarm enable/disable (Boolean)

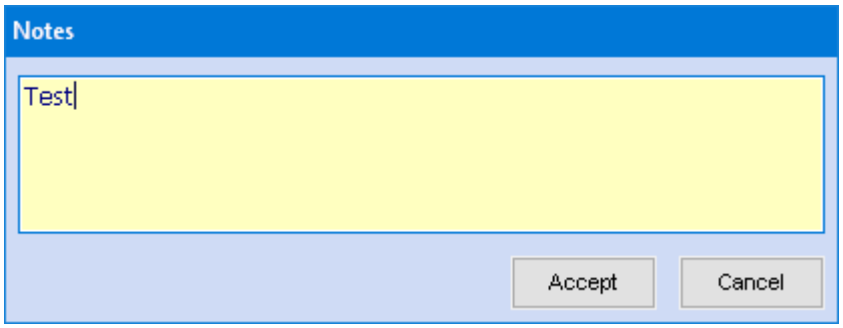
Description	SetAlarmPar (AlarmName, AlarmValue) function sets alarm value (database transaction only) and returns TRUE if transaction is successful or FALSE if transaction is NOT successful
Syntax	Project. SetAlarmPar m_return = Project. SetAlarmPar (AlarmName, "10:60")
Parameter(s)	AlarmName – Alarm name (string) AlarmValue – Alarm preset value

Description	RequestAlarmsUpdate () method commits changes to ClearView-SCADA Server
Syntax	Project. RequestAlarmsUpdate
Parameter(s)	None

Description	ViewReport “Report Name” opens built-in or custom report viewer
Syntax	Project. ViewReport “ReportName”
Parameter(s)	Report name: For built-in reports one of: "Alarm Log" "Audit Trail Log" "Security Log" "Lockout/Tagout Log" "Data Log" "Users Log" For custom reports: "Custom Reports"

Description	CustomEvent “Event Message” creates user defined event and adds this event to the database
Syntax	Project.CustomEvent “YourMessage”
Parameter(s)	YourMessage -String

Description	CustomEvent “Event Message” creates user defined event and adds this event to the database
Syntax	Project.CustomEvent “YourMessage”
Parameter(s)	YourMessage -String

Description	ViewNotes "NoteTag", 1 opens/views user defined notes
Syntax	Project.ViewNotes "NoteTag", 1 
Parameter(s)	NoteTag (String) – specify variable tag to which the note would be assigned 0 – Opens note interface for view only 1 – Opens for view and edit

Description	ControlItemArray (ControlItem0, 2) - function to covert user defined ControlItem string in to Array and access by Array Index
Syntax	Example: ButtonCtrl3.ControlItem0 = "test1,test2,test3,test4" Msgbox Project.ControlItemArray (ButtonCtrl3.ControlItem0, 2)
Parameter(s)	ButtonCtrl3.ControlItem0 – String with comma "," separator. 2 – Array Index (return: "test3")

Function	Description	Parameters	Example
Project.CopyFile	Copy a file.	Source (String), Destination (String) OverWriteFiles (Boolean)	*mReturn = Project.CopyFile ("C:\Test_Files\Folder1\Test1.csv", "C:\Test_Files\Folder2\", 1)* *Application.MsgBox "Retrun: " & mReturn*
Project.CopyFolder	Copy a folder.	Source (String), Destination (String) OverWriteFiles (Boolean)	*mReturn = Project.CopyFolder ("C:\Test_Files\Folder1", "C:\Test_Files\Folder2", 1)* *Application.MsgBox "Retrun: " & mReturn*
Project.CreateFolder	Create a folder.	Source (String)	*mReturn = Project.CreateFolder ("C:\Test_Files\Folder1")* *Application.MsgBox "Retrun: " & mReturn*
Project.DeleteFile	Delete a file.	Source (String) Force (Boolean)	*mReturn = Project.DeleteFile ("C:\Test_Files\Folder1\Test0.csv", 1)* *Application.MsgBox "Retrun: " & mReturn*

Function	Description	Parameters	Example
Project.DeleteFolder	Delete a folder.	Source (String) Force (Boolean)	*mReturn = Project.DeleteFolder ("C:\Test_Files\Folder2", 1)* *Application.MsgBox "Retrun: " & mReturn*
Project.FileExists	Check if a file exists.	Source (String)	*mReturn = Project.FileExists ("C:\Test_Files\Folder1\Test0.csv")* *Application.MsgBox "Retrun: " & mReturn*
Project.FolderExists	Check if a folder exists.	Source (String)	*mReturn = Project.FolderExists ("C:\Test_Files\Folder1")* *Application.MsgBox "Retrun: " & mReturn*
Project.MoveFile	Move a file.	Source (String) Destination (String)	*for i = 0 to 9* *mReturn = Project.MoveFile ("C:\Test_Files\Folder2\Test" & i & ".csv", "C:\Test_Files\Folder1\")* *next* *Application.MsgBox "Retrun: " & mReturn*
Project.MoveFolder	Move a folder.	Source (String) Destination (String)	*mReturn = Project.MoveFolder ("C:\Test_Files\Folder1", "C:\Test_Files\Folder2")* *Application.MsgBox "Retrun: " & mReturn*
Project.CreateFile	Create a file.	Destination (String) OverWriteFiles (Boolean)	*for i = 0 to 9* *mReturn = Project.CreateFile ("C:\Test_Files\Folder1\Test" & i & ".csv", 1)* *next* *Application.MsgBox "Retrun: " & mReturn*
Project.WriteFile	Open a file for writing.	Source (String) TxtLine (String) Mode (Integer) -Mode = 0 - Writing -Mode = 1 – Appending Create (Boolean)	*for i = 0 to 99* *mReturn = Project.WriteFile ("C:\Test_Files\Folder1\Test1.csv", cstr("Row" & i), 1, 0)* *next* *Application.MsgBox "Retrun: " & mReturn* *'Note: If Create set to True (1) the new file is created*
Project.ReadFile	Open a file for reading.	Source (String) TxtLine (Array)	*mReturn = Project.ReadFile ("C:\Test_Files\Folder1\Test1.csv", Line)* *Application.MsgBox "Retrun: " & mReturn* *For i = 0 to Ubound (Line) - 1* *ListBox.AddItem Line (i)* *Next*
Project.PrintFile	Prints existing file.	Source (String)	*mReturn = Project.PrintFile ("C:\Test_Files\Folder1\Test1.csv")* *Application.MsgBox "Retrun: " & mReturn*

Note: Functions returns error description or OK status – string

Description	***SelectHistory (TagName, StartTime, EndTime, ArrayofTimes, ArrayofValues, Optional MinValue, Optional MaxValue)*** function quiries History table and returns values for specified period of time.
Syntax	***Project. SelectHistory*** Private Sub Button_Click( ) Ex_ListBox1.Clear *iRetrurn = Project.SelectHistory (TagName, StartTime, EndTime, ArrayofTimes, ArrayofValues, MinValue, MaxValue, AVGValue)* For i = 0 to iRetrurn ListBox.AddItem CDate(ArrayofTimes(i)) & " ----------" & ArrayofValues(i) Next MsgBox MinValue & "----------" & MaxValue End Sub
Parameter(s)	TagName – Name of the tag you would like to quiery (String) StartTime – Start time parameter (Double Float) EndTime – End time parameter (Double Float) ArrayofTimes – Returns Array of timestamps for specified time period (Array of Double Floats) ArrayofValues – Returns Array of values for specified time period (Array of Double Floats) MinValue – Returns minimum value for specified time period (Double Floats) MaxValue – Returns maximum value for specified time period (Double Floats) AVGValue – Returns average value for specified time period (Double Floats) iReturn – Function returns number of records (Long)

Events

Description	Project_OnGISDblClick(LocationName) event occurs on mouse double-click on GIS location or junction
Syntax	Private Sub Project_OnGISDblClick(LocationName) 'Your Code Here End Sub
Parameter(s)	LocationName – String: the name of location being clicked

Description	Project_OnGISDblClickEx(LocationName, MapName, Device, Custom) event occurs on mouse double-click on GIS location or junction
Syntax	Private Sub Project_OnGISDblClickEx(LocationName, MapName, Device, Custom) 'Your Code Here End Sub
Parameter(s)	LocationName – String: the name of location clicked MapName – String: the name of the map clicked Device – String: "Device" property of location clicked Custom – String: "Custom" property of location clicked

Description	Project_ExitProjectEvent() event occurs on project or ClearView exit
Syntax	Private Sub Project_ExitProjectEvent( ) 'Your Code Here End Sub
Parameter(s)	None

Security Object

Properties

Description	IsLoggedIn property gets Boolean value indicating if the ClearView User is Logged In

Syntax	‘Security.IsLoggedIn
	Sub Pad_CCPadUpdate( )
	  If Security.IsLoggedIn= True Then
	  ‘Your Code Here
	  End If
	End Sub

Data Type	Boolean
Environment	Run Time
Access	Read Only

Description	CurrentUser property gets ClearView Logged in User’s Name

Syntax	‘Security.CurrentUser
	Sub Object1_Click ( )
	  If Security.CurrentUser= “User Name” Then
	  ‘Your Code Here
	  End If
	End Sub

Data Type	String
Environment	Run Time
Access	Read Only

Methods

Description	Check (Zone, Reserved, ShowMsg) method checks Zone Permissions, return Boolean (True/False). If the current user has permissions to the Zone the Security.Check will return True , otherwise - False

Syntax	‘Security.Check
	Private Sub Object1_Click( )
	If Security.Check(“1”,,0) = True Then
	“Your Code Here”
	End If
	End Sub

Parameter(s)	Zone-String or Number: Security zone Reserved: not used, should be blank ShowMsg - Number: 0 – Show Message Box, 1 – Don’t show Message Box

Description	Login [UserName, Password] method logs in a user. Using the Login method without parameters will open user login screen, using the Login method with parameters will login the specified user
Syntax	‘Using the Login method without parameters will open user login screen Private Sub Object1_Click( ) Security.Login End Sub ‘Using the Login method with parameters will login the specified user Private Sub Object1_Click( ) Security.Login “Admin”, “ClearView” End Sub
Parameter(s)	UserName – String: User Name Password – String: User password

Description	LogOut method logout current user
Syntax	‘Security.LogOut Private Sub Object1_Click( ) Security.LogOut End Sub
Parameter(s)	None

Events

Description	Security_LoggedIn () event occurs on user login
Syntax	‘Security_LoggedIn () Private Sub Security_LoggedIn() ‘Your Code Here End Sub
Parameter(s)	None

Description	Security_LoggedOut () event occurs on user logout
Syntax	‘Security_LoggedOut () Private Sub Security_LoggedOut() ‘Your Code Here End Sub
Parameter(s)	None

Tag Object

Properties

Description	AuditTrailed property gets the Boolean value indicating if the tag's Audit Trail is set, returns True if it is set and False otherwise

Syntax	'Tags.item("Your_Tag_Name").AuditTrailed
	Private Sub Object1_Click( )
	  If Tags.item("Your_Tag_Name").AuditTrailed = True Then
	  'Your Code Here
	  End If
	End Sub

Data Type	Boolean
Environment	Run Time
Access	Read Only

Description	ClampHigh property gets the Boolean value indicating if the tag's "Clamp High" is set (True/False)

Syntax	'Tags.item("Your_Tag_Name").ClampHigh
	Private Sub Object1_Click( )
	  If Tags.item("Your_Tag_Name").ClampHigh = True Then
	  'Your Code Here
	  End If
	End Sub

Data Type	Boolean
Environment	Run Time
Access	Read Only

Description	ClampLow property gets the Boolean value indicating if the tag's "Clamp Low" is set (True/False)

Syntax	'Tags.item("Your_Tag_Name").ClampLow
	Private Sub Object1_Click( )
	  If Tags.item("Your_Tag_Name").ClampLow = True Then
	  'Your Code Here
	  End If
	End Sub

Data Type	Boolean
Environment	Run Time
Access	Read Only

Description	DeadBand property gets tag's Dead Band Value

Syntax	'Tags.item("Your_Tag_Name").DeadBand
	Private Sub Object1_Click( )
	  If Tags.item("Your_Tag_Name").Deadband >= 100 Then
	  'Your Code Here
	  End If
	End Sub

Data Type	Integer
Environment	Run Time
Access	Read Only

Description	Description property gets Tag Description

Syntax	‘Tags.item("Your_Tag_Name").Description
	
	Private Sub Object1_Click( )
	  If Tags.item("Your_Tag_Name").Description = “Your_Tag_Name” Then
	  ‘Your Code Here
	  End If
	End Sub

Data Type	String
Environment	Run Time
Access	Read Only

Description	Error property gets OPC Tag Error

Syntax	‘Tags.item("Your_Tag_Name").Error
	Private Sub Object1_Click( )
	  If Tags.item("Your_Tag_Name").Error > 0 Then
	  ‘Your Code Here
	  End If
	End Sub

Data Type	Long
Environment	Run Time
Access	Read Only

Description	EventLogged property gets the Boolean value indicating if the tag is using Event Logging (True/False)

Syntax	‘Tags.item("Your_Tag_Name").EventLogged
	Private Sub Object1_Click( )
	  If Tags.item("Your_Tag_Name"). EventLogged = True Then
	  ‘Your Code Here
	  End If
	End Sub

Data Type	Boolean
Environment	Run Time
Access	Read Only

Description	EventLogTag property gets Event Log tag

Syntax	‘Tags.item("Your_Tag_Name").EventLogTag
	Private Sub Object1_Click( )
	  If Tags.item("Your_Tag_Name").EventLogTag = "Your_Tag_Name1" Then
	  ‘Your Code Here
	  End If
	End Sub

Data Type	String
Environment	Run Time
Access	Read Only

Description	Expression property gets the expression for Derived Tag , applicable to derived tags only

Syntax	‘Tags.item("Your_Tag_Name").Expression
	Private Sub Object1_Click( )
	  If Tags.item("Your_Tag_Name").Expression = "Your_Text" Then
	  ‘Your Code Here
	  End If
	End Sub

Data Type	String
Environment	Run Time
Access	Read Only

Description	Item property gets OPC Item path, for OPC Tag only

Syntax	‘Tags.item("Your_Tag_Name").Item
	Private Sub Object1_Click( )
	  If Tags.item("Your_Tag_Name").Item = "Your_Text" Then
	  ‘Your Code Here
	  End If
	End Sub

Data Type	String
Environment	Run Time
Access	Read Only

Description	Logged property gets the Boolean value indicating if the tag is Logged (True/False)

Syntax	‘Tags.item("Your_Tag_Name").Logged
	Private Sub Object1_Click( )
	  If Tags.item("Your_Tag_Name").Logged = True Then
	  ‘Your Code Here
	  End If
	End Sub

Data Type	Boolean
Environment	Run Time
Access	Read Only

Description	LogInterval property gets the Logging Interval in seconds

Syntax	‘Tags.item("Your_Tag_Name").LogInterval
	Private Sub Object1_Click( )
	  If Tags.item("Your_Tag_Name").LogInterval >= 60 Then
	  ‘Your Code Here
	  End If
	End Sub

Data Type	Long
Environment	Run Time
Access	Read Only

Description	Node property gets OPC Server Node Name

Syntax	‘Tags.item("Your_Tag_Name").Node
	Private Sub Object1_Click( )
	  If Tags.item("Your_Tag_Name").Node = “Your String Here” Then
	  ‘Your Code Here
	  End If
	End Sub

Data Type	String
Environment	Run Time
Access	Read Only

Description	ProgID property gets OPC Server Program ID

Syntax	‘Tags.item("Your_Tag_Name").ProgID
	
	Private Sub Object1_Click( )
	  If Tags.item("Your_Tag_Name").ProgID = “Your String Here” Then
	  ‘Your Code Here
	  End If
	End Sub

Data Type	String
Environment	Run Time
Access	Read Only

Description	Quality property gets OPC Tag Quality. Returns 192 for Good Quality

Syntax	‘Tags.item("Your_Tag_Name").Quality
	Private Sub Object1_Click( )
	  If Tags.item("Your_Tag_Name").Quality <> 192 Then
	  ‘Your Code Here
	  End If
	End Sub

Data Type	Long
Environment	Run Time
Access	Read Only

Description	RawMax property gets OPC Tag Raw Maximum Scaling

Syntax	‘Tags.item("Your_Tag_Name").RawMax
	
	Private Sub Object1_Click( )
	  If Tags.item("Your_Tag_Name").RawMax <> 999.999 Then
	  ‘Your Code Here
	  End If
	End Sub

Data Type	Double
Environment	Run Time
Access	Read Only

Description	RawMin property gets OPC Tag Raw Minimum Scaling

Syntax	‘Tags.item("Your_Tag_Name").RawMin
	
	Private Sub Object1_Click( )
	  If Tags.item("Your_Tag_Name").RawMin <> 100.777 Then
	  ‘Your Code Here
	  End If
	End Sub

Data Type	Double
Environment	Run Time
Access	Read Only

Description	ScaleMax property gets OPC Tag Scaled Maximum

Syntax	‘Tags.item("Your_Tag_Name").ScaleMax
	
	Private Sub Object1_Click( )
	  If Tags.item("Your_Tag_Name").ScaleMax <> 100 Then
	  ‘Your Code Here
	  End If
	End Sub

Data Type	Double
Environment	Run Time
Access	Read Only

Description	ScaleMin property gets OPC Tag Scaled Minimum

Syntax	‘Tags.item("Your_Tag_Name").ScaleMin
	Private Sub Object1_Click( )
	  If Tags.item("Your_Tag_Name").ScaleMin < 1 Then
	  ‘Your Code Here
	  End If
	End Sub

Data Type	Double
Environment	Run Time
Access	Read Only

Description	Tag property gets Tag Name

Syntax	‘Tags.item(Your_Tag_ID).Tag
	‘ The Tag Name is retrieved based on TagID (index)
	Private Sub Object1_Click( )
	  If Tags.item(2).Tag = “Your Tag Name” Then
	  ‘Your Code Here
	  End If
	End Sub

Data Type	String
Environment	Run Time
Access	Read Only

Description	TagID property gets Tag ID

Syntax	‘Tags.item(Your_Tag_Name).TagID
	Private Sub Object1_Click( )
	  If Tags.item(“Your_Tag_Name”).TagID = 2 Then
	  ‘Your Code Here
	  End If
	End Sub

Data Type	Long
Environment	Run Time
Access	Read Only

Description	TagScale property gets the Boolean value indicating if OPC Tag Scaling is applied (True/False)

Syntax	‘Tags.item(Your_Tag_Name).TagScale
	
	Private Sub Object1_Click( )
	  If Tags.item(“Your_Tag_Name”).TagScale = True Then
	  ‘Your Code Here
	  End If
	End Sub

Data Type	Boolean
Environment	Run Time
Access	Read Only

Description	TagType property gets Tag Type. OPC Tag = 0, Derived Tag = 1, Variable Tag = 2

Syntax	‘Tags.item(Your_Tag_Name).TagType
	Private Sub Object1_Click( )
	  If Tags.item(“Your_Tag_Name”).TagType = 2 Then
	  ‘Your Code Here
	  End If
	End Sub

Data Type	Integer
Environment	Run Time
Access	Read Only

Description	Trended property gets the Boolean value indicating if the Tag is Trended (True/False)

Syntax	‘Tags.item(Your_Tag_Name).Trended
	Private Sub Object1_Click( )
	  If Tags.item(“Your_Tag_Name”).Trended = True Then
	  ‘Your Code Here
	  End If
	End Sub

Data Type	Boolean
Environment	Run Time
Access	Read Only

Description	Value property gets and sets the Tag Value

Syntax	‘Tags.item(Your_Tag_Name).Value
	Private Sub Object1_Click( )
	  If Tags.item(“Your_Tag_Name1”).Value = 100 Then
	  Tags.item(“Your_Tag_Name2”).Value = 0
	  End If
	End Sub

Data Type	Variant
Environment	Run Time
Access	Read/Write

Methods

None.

Events

Description	Tags_BadQuality event occurs when tag’s quality is changed to “bad”

Syntax	‘Private Sub Tags_BadQuality
	Private Sub Tags_BadQuality( )
	  If Tags.item(“Your_Tag_Name”).Quality <> 192 Then
	  ‘Your Code Here
	  End If
	End Sub

Parameter(s)	None

Description	Tags_Changed (Count, Tag, Value, Quality, Timestamp) event occurs on Tag Change (Update)
Syntax	‘Adding tag changes to a Listbox Private Sub Tags_TagChanged(Count, Tag, Value, Quality, Timestamp) For i = 1 To Count ListBoxCtrl1.AddItem "Tag = " & Tag(i) & " Value = " & Value(i) & " Quality = " & Quality(i) & " Timestamp = " & Timestamp(i) Next End Sub
Parameter(s)	Count – number of changes Tag – array of tag names Value – array of tag values Quality – array of tag qualities Timestamp – array of tag timestamps

SubscribedAlarms Object

Warning: SubscribedAlarms' Properties and Methods can be used only if there are subscribed alarm groups with alarms configured. In case of no subscribed alarms the Methods and Properties below will generate errors.

Properties

Description	Count property gets the number of subscribed alarms

Syntax	'Cycle through Subscribed Alarms
	Private Sub Object1_Click( )
	For alarm = 1 to SubscribedAlarms.Count
	If SubscribedAlarms.item(alarm).Acknowledged = True Then
	'Your Code Here
	End If
	Next
	End Sub

Data Type	Integer
Environment	Run Time
Access	Read Only

Description	Acknowledged property gets the Boolean value indicating if the Alarm is Acknowledged (True/False)

Syntax	'SubscribedAlarms.item("Your_Alarm_Name").Acknowledged
	Private Sub Object1_Click( )
	If SubscribedAlarms.item("Your_Alarm_Name").Acknowledged = True Then
	'Your Code Here
	End If
	End Sub

Data Type	Boolean
Environment	Run Time
Access	Read Only

Description	AcknowledgedTime property gets Alarm Acknowledged Timestamp
Syntax	‘SubscribedAlarms.item("Your_Alarm_Name").AcknowledgedTime ACKTime = SubscribedAlarms.item("Your_Alarm_Name").AcknowledgedTime
Data Type	Double
Environment	Run Time
Access	Read Only

Description	Active property gets the Boolean value indicating if the Alarm is Active (True/False)

Syntax	‘SubscribedAlarms.item("Your_Alarm_Name").Active
	Private Sub Object1_Click( )
	  If SubscribedAlarms.item("Your_Alarm_Name").Active = True Then
	  ‘Your Code Here
	  End If
	End Sub

Data Type	Boolean
Environment	Run Time
Access	Read Only

Description	ActiveTime property gets Alarm Active Timestamp
Syntax	‘SubscribedAlarms.item("Your_Alarm_Name").ActiveTime ActiveTime = SubscribedAlarms.item("Your_Alarm_Name").ActiveTime
Data Type	Double
Environment	Run Time
Access	Read Only

Description	ClearedTime property gets Alarm Cleared Timestamp
Syntax	‘SubscribedAlarms.item("Your_Alarm_Name").ClearedTime ClearedTime = SubscribedAlarms.item("Your_Alarm_Name").ClearedTime
Data Type	Double
Environment	Run Time
Access	Read Only

Description	State property gets Alarm State 0 - No Alarm 1 - Cleared and Unacknowledged 2 - Active and Acknowledged 3 - Active and Unacknowledged

Syntax	‘SubscribedAlarms.item("Your_Alarm_Name").State
	Private Sub Object1_Click( )
	  If SubscribedAlarms.item("Your_Alarm_Name").State = 0 Then
	  ‘Your Code Here
	  End If
	End Sub

Data Type	Integer
Environment	Run Time
Access	Read Only

Description	AlarmConfig.AlarmConditionValue property gets specified alarm condition value
Syntax	‘Gets Alarm_Name condition value cVal = SubscribedAlarms.item("Alarm_Name").AlarmConfig.AlarmConditionValue End Sub
Data Type	String
Environment	Run Time
Access	Read Only

Description	AlarmConfig.AlarmConditionDisplay property gets the entire condition in plain text

Syntax	‘SubscribedAlarms.item("Your_Alarm_Name").AlarmConfig.AlarmConditionDisplay
	Private Sub Object1_Click( )
	If SubscribedAlarms.item("Alarm ").AlarmConfig.AlarmConditionDisplay = “Sim1 not in range (1:3)“ Then
	‘Your Code Here
	End If
	End Sub

Data Type	String
Environment	Run Time
Access	Read Only

Description	AlarmConditionOperator property gets Alarm Condition Operator. Returns: 1 – if Alarm condition is set to “>“ 2 – “<” 3 – “>=” 4 – “<=” 5 – “=” 6 – “<>” 7 – “in Range” 8 – “not in range”

Syntax	‘SubscribedAlarms.item("Your_Alarm_Name").AlarmConditionOperator
	Private Sub Object1_Click( )
	  If SubscribedAlarms.item("Alarm_Name").AlarmConditionOperator = 5 Then
	  ‘Your Code Here
	  End If
	End Sub

Data Type	Integer
Environment	Run Time
Access	Read Only

Description	AlarmConditionTag property gets Alarm Condition Tag

Syntax	‘SubscribedAlarms.item("Your_Alarm_Name").AlarmConditionTag
	Private Sub Object1_Click( )
	  If SubscribedAlarms.item("Alarm_Name").AlarmConditionTag = “Tag_Name“ Then
	  ‘Your Code Here
	  End If
	End Sub

Data Type	String
Environment	Run Time
Access	Read Only

Description	AlarmConditionType property gets Alarm Condition Type. Returns: 1 – Alarm when <tag> is On, 2 – Alarm when <tag> is Off, 3 – Alarm on <value>

Syntax	‘SubscribedAlarms.item("Your_Alarm_Name").AlarmConditionType
	Private Sub Object1_Click( )
	  If SubscribedAlarms.item("Alarm_Name").AlarmConditionType = 2 Then
	  ‘Your Code Here
	  End If
	End Sub

Data Type	Integer
Environment	Run Time
Access	Read Only

Description	AlarmDescription property gets Alarm Description

Syntax	‘SubscribedAlarms.item("Your_Alarm_Name").AlarmDescription
	Private Sub Object1_Click( )
	  If SubscribedAlarms.item("Alarm_Name").AlarmDescription= “Text” Then
	  ‘Your Code Here
	  End If
	End Sub

Data Type	String
Environment	Run Time
Access	Read Only

Description	AlarmGroup property gets Alarm Group the alarm belongs to

Syntax	‘SubscribedAlarms.item("Your_Alarm_Name").AlarmGroup
	Private Sub Object1_Click( )
	  If SubscribedAlarms.item("Alarm_Name").AlarmGroup= “Text” Then
	  ‘Your Code Here
	  End If
	End Sub

Data Type	String
Environment	Run Time
Access	Read Only

Description	AlarmName property gets Alarm Name (based on Alarm ID)

Syntax	‘SubscribedAlarms.item(alarm_ID).AlarmName
	Private Sub Object1_Click( )
	  If SubscribedAlarms.item(1).AlarmName= “Your Alarm Name” Then
	  ‘Your Code Here
	  End If
	End Sub

Data Type	String
Environment	Run Time
Access	Read Only

Description	Emailed property Returns if Alarm emailed is specified (0/1)

Syntax	‘SubscribedAlarms.item("Your_Alarm_Name").Emailed
	
	Private Sub Object1_Click( )
	  If SubscribedAlarms.item("Alarm_Name").Emailed = 1 Then
	  ‘Your Code Here
	  End If
	End Sub

Data Type	Integer
Environment	Run Time
Access	Read Only

Description	EmailTo property gets email address recipients

Syntax	‘SubscribedAlarms.item("Your_Alarm_Name").EmailTo
	
	Private Sub Object1_Click( )
	  If SubscribedAlarms.item("Alarm_Name").EmailTo= “email@email.com” Then
	  ‘Your Code Here
	  End If
	End Sub

Data Type	String
Environment	Run Time
Access	Read Only

Description	Severity property gets Alarm Severity

Syntax	‘SubscribedAlarms.item("Your_Alarm_Name").Severity
	Private Sub Object1_Click( )
	  If SubscribedAlarms.item("Alarm_Name").Severity= 2 Then
	  ‘Your Code Here
	  End If
	End Sub

Data Type	Integer
Environment	Run Time
Access	Read Only

Methods

Description	AcknowledgeAll method Acknowledges All Alarms
Syntax	‘SubscribedAlarms.AcknowledgeAll Private Sub Object1_Click( ) SubscribedAlarms.AcknowledgeAll End Sub
Parameter(s)	N/A

Description	Acknowledge method Acknowledges Specified Alarm
Syntax	‘SubscribedAlarms.Item("Your_Alarm_Name").Acknowledge Private Sub Object1_Click( ) SubscribedAlarms.Item("Your_Alarm_Name").Acknowledge End Sub
Parameter(s)	N/A

Events

Description	AlarmConfigChanged event Occurs when alarm configuration changed
Syntax	‘Private Sub SubscribedAlarms_AlarmConfigChanged( ) Private Sub SubscribedAlarms_AlarmConfigChanged( ) ‘Your Code Here End Sub
Parameter(s)	N/A

Description	AlarmStateChanged (AlarmID, OldState, OldIsReset) event occurs when alarm state changes

Syntax	‘Private Sub SubscribedAlarms_AlarmStateChanged (AlarmID, OldState, OldIsReset)
	Private Sub SubscribedAlarms_AlarmStateChanged(AlarmID, OldState, OldIsReset)
	  If AlarmID = 1 And OldState = 3 Then
	  ‘Your Code Here
	  End If
	End Sub

Parameter(s)	AlarmID – Alarm Index (Alarm Name can be retrieved based on AlarmID) OldState – The previous Alarm State. Alarmstates are: 0 - No Alarm 1 - Cleared and Unacknowledged 2 - Active and Acknowledged 3 - Active and Unacknowledged 4 – Reset OldIsReset - Boolean: True if previously the alarm was reset or False if alarm wasn’t reset

Description	AlarmStatusChanged (AlarmName, CurrentState) event Occurs when alarm status changes

Syntax	‘SubscribedAlarms_AlarmStatusChanged(AlarmName, CurrentState)
	Private Sub SubscribedAlarms_ AlarmStatusChanged(AlarmName, CurrentState)
	  If AlarmName = “Your Alarm Name” And CurrentState = 3 Then
	  ‘Your Code Here
	  End If
	End Sub

Parameter(s)	AlarmName – Alarm Name CurrentState – The current Alarm State

Debug Object

The debug object is used to troubleshoot the syntax in run-time. The debug will print errors into debug window.

Properties

Description	Enable property sets or gets the Debugging mode (True/False). Debugging output goes to the “Debug” window that can be opened through Menu: **View->Debug Window**

Syntax	‘Debug.Enable
	Private Sub Object1_Click( )
	  If Debug.Enable = False Then
	  Debug.Enable = True
	  End If
	End Sub

Data Type	Boolean
Environment	Run Time
Access	Read/Write

Methods

Description	SetMaxLines (lines) method sets Debug Print Maximum Lines Limit
Syntax	‘Set number of lines to 50 Private Sub Object1_Click( ) Debug.SetMaxLines 50 End Sub
Parameter(s)	Lines – Integer: number of lines

Description	PrintMsg (Message) method prints Debug Message –
Syntax	‘Print Debug message Private Sub Object1_Click( ) Debug.PrintMsg “Your Object Here” End Sub
Parameter(s)	Message – String: message to print

Events

None.

LocalMemory Object

Local Memory is a GLOBAL variable for ClearView client. The variable can be access by any ClearView screen scripting (ClearScript). The LocalMemory functions are very useful for passing references of tags, alarms or security settings.

Properties

None.

Methods

Description	Save Value, Name method saves variable value by name
Syntax	‘LocalMemory.Save Value, Name Private Sub Object1_Click( ) LocalMemory.Save 100, “My_Variable_Name” End Sub
Parameter(s)	Name – The Variable Name Value – The Variable Value

Description	Value (Name) Retrieves variable value by name

Syntax	‘LocalMemory.Value
	
	Private Sub Object1_Click( )
	If LocalMemory.Value (“My_Variable_Name”) = 100 Then
	  ‘Your Code Here
	End If
	End Sub

Parameter(s)	Name – The Variable Name

Events

None.

Blink Object

There are four Blink objects in ClearView: Blink_F, Blink_FF, Blink_S, Blink_SS. The objects change “Interval” property between True and False in configured time intervals.

Note: choose the Blink object that has time interval that is less than CCPad update interval.

Default time intervals for Blink_XX objects are:

Blink_F - Blink Fast 250 milliseconds

Blink_FF – Blink Fast Fast 100 milliseconds

Blink_S – Blink Slow 500 milliseconds

Blink_SS – Blink Slow Slow 1000 milliseconds

Properties

Description	Interval property returns True or False

Syntax	‘Switching (Blinking) object’s color between Red and Black if tag’s quality is bad
	If Tags.item("Your_Tag_Name").Quality <> 192
	  If Blink_SS.Interval = True Then
	  Object1.BackColor = VBRed
	  else
	  Object1.BackColor = VBBlack
	  End If
	End if

Parameter(s)	Key – The Variable Name

Methods

None

Events

None

ActiveX Extended Properties

Property Name	Description	Data Type	Access
Left	Returns/Sets left poison (in pixels) of the control	Long	Read/Write
Top	Returns/Sets top poison (in pixels) of the control	Long	Read/Write
Height	Returns/Sets pixel Height of the control	Long	Read/Write
Visible	Returns/Sets False/True if the control is visible at runtime	Boolean	Read/Write
Tag	Returns/Sets any extra data for your application	String	Read/Write
TabStop	Control can receive keyboard input focus	Boolean	None
TabIndex	Tab Index of the control	Integer	None
ToolTipText	Returns/Sets the text displayed when the mouse is paused over the control	String	Read/Write
ControlFrame	Returns/Sets Control Mouse Over frame style	Integer	Read/Write
ControlFrameColor	Returns/Sets Control Mouse Over frame Color	OLE Color	Read/Write
ControlSecurity	Returns/Sets a value indicating the security level required to use the control	Integer	Read/Write
ControlItem0	Returns/Sets any extra data for your application	String	Read/Write
ControlItem1	Returns/Sets any extra data for your application	String	Read/Write
ControlItem2	Returns/Sets any extra data for your application	String	Read/Write
ControlItem3	Returns/Sets any extra data for your application	String	Read/Write
ControlItem4	Returns/Sets any extra data for your application	String	Read/Write
ControlItem5	Returns/Sets any extra data for your application	String	Read/Write
ControlItem6	Returns/Sets any extra data for your application	String	Read/Write
ControlItem7	Returns/Sets any extra data for your application	String	Read/Write
ControlItem8	Returns/Sets any extra data for your application	String	Read/Write
ControlItem9	Returns/Sets any extra data for your application	String	Read/Write

ClearView Scripting Examples

Security

```
Code:
Private Sub Object1_Click( )
'Logout If pipe clicked
'Display Isloggedin property value
    Security.LogOut
    MsgBox Security.IsLoggedIn
End Sub
'-------------------------------------
Sub Object1_CCPadUpdate( )
'On update check logged in value,
'If False Then prompt user To login
    If Not Security.IsLoggedIn Then
        Security.Login
    End If

End Sub
```

Result: Clicking on the Object_1 logged the current user off.

Tags

```
Code:
Private Sub Object1_Click( )
Dim TagDesc
    TagDesc = Tags.Item("Blink").Description
    MsgBox TagDesc
Dim TagVal
    TagVal = Tags.Item("Blink").value
    MsgBox TagVal
    Tags.Item("Test").value = 99
End Sub
```

Result: The tag by the name of Test value changed to 99.

Subscribed Alarms

```
Code:
Dim IsActive
Dim myAlarmState
Dim myAcknowledeged
Private Sub Object1_Click( )
'Notice the use of the alarm name test instead of a numerical index
    myAlarmState = SubscribedAlarms.Item("TEST").state
    MsgBox myAlarmState

    IsActive = SubscribedAlarms.Item(1).active
    MsgBox IsActive

    myAcknowledged = SubscribedAlarms.Item(1).acknowledged
    MsgBox myAcknowledged

End Sub
```

Result: The alarm state message box is shown. It displays a value of 0, which indicates there is no alarm. Also, note the Alarm Group "TEST" is checked – which is required in order to access the alarm properties. Two more message boxes will follow the state message box.

Screens

```
Code:


Private Sub Object1_Click( )
    Screens.ShowScreen(NonExistent)
'Note the use of quotes
    Screens.ShowScreen("MyScreen")
    Screens.ShowScreen("EventView")
    Screens.CloseScreen("MyScreen")

End Sub
```

Result: The first call to ShowScreen uses a name without quotes and a name that is not a screen or a ClearView form. This call results in an error.

Local Memory

Code :

```
Private Sub Object1_Click( )
    LocalMemory.Save 100, "MyString"
    MsgBox LocalMemory.Value("MyString")
End Sub
```

Result: After running the LocalMemory.Save function, LocalMemory.Value can be used from any screen to retrieve this value (100) using the key, Mystring.

ActiveX Object Example

Code :

```
Sub Object1_CCPadUpdate( )
    If Tags.Item("Blink").Value = 0 Then
        Object1.BackColor = vbYellow
        Object1.FalseFillColor = vbYellow
    Else
        Object1.BackColor = vbBlack
        Object1.FalseFillColor = vbBlack
    End If
End Sub
```

Or

```
Sub Object1_CCPadUpdate( )
    If Second(Now()) Mod 2 = 0 Then
        Object1.BackColor = vbYellow
        Object1.FalseFillColor = vbYellow
    Else
        Object1.BackColor = vbBlack
        Object1.FalseFillColor = vbBlack
    End If
End Sub
```

Result: This code utilizes an existing system tag, Blink, which the server sets on/off every second or example to achieve the same result without the system tag - Blink

Server Scripting

Server scripting is convenient when a script needs to run all the time. For example, after a computer is restarted and no one has yet logged in, the ClearView client cannot be started and cannot run a script. This is called client-side scripting. Alternatively, the script running on the ClearView Server has the ability to run as a process in the background and starts regardless of a user logging into the computer. This is called server-side scripting.

The script is created in the ClearView client using the Server Scripting Editor. The editor can be accessed via the Application Explorer in the Scripting folder.

Double-clicking **Server Scripting** will launch the Server Scripting Editor.

After you have created a script and saved the VBS file, you can exit the Server Scripting Editor and return to the ClearView client.

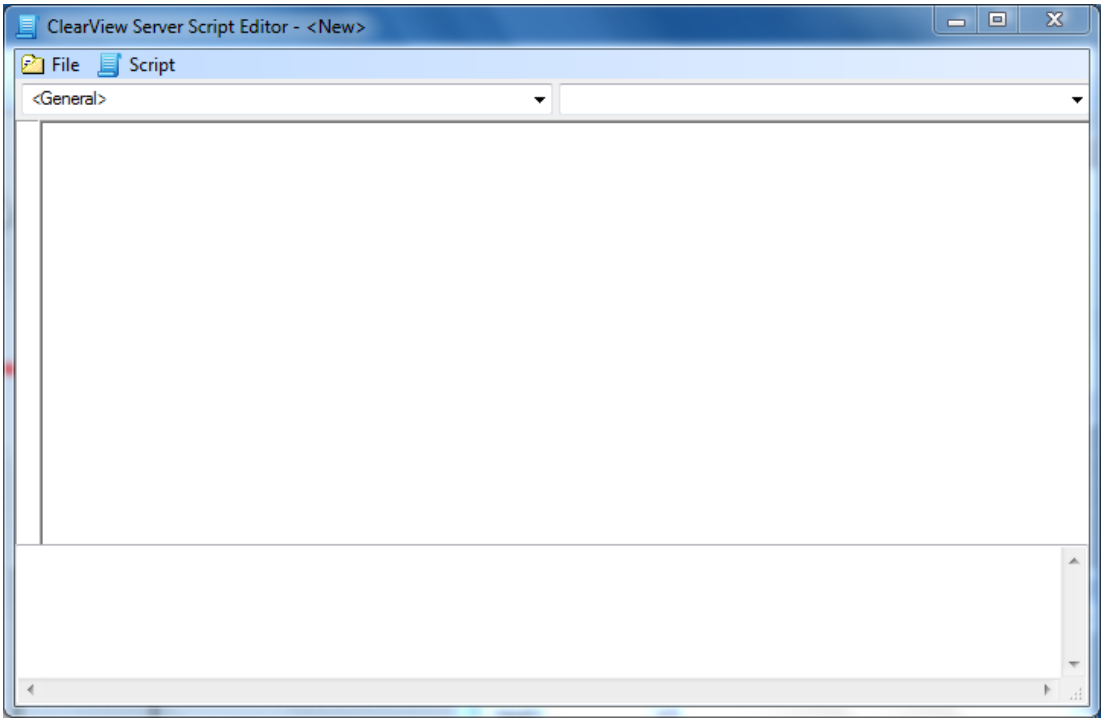
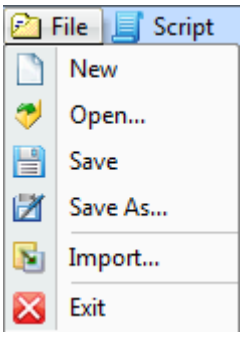
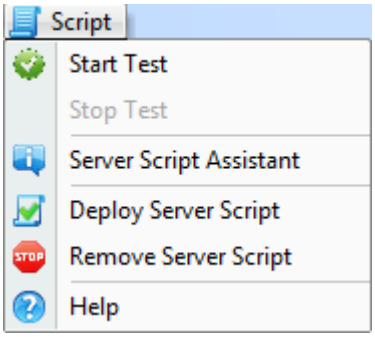


Figure #10.5

Server Script Editor Menu

Menu	Menu Item	Description
	New	Specify a new VBS file.
	Open	Open a VBS file.
	Save	Save the VBS file.
	Save As	Specify a location to save the VBS file.
	Import	Imports the legacy Server Script configuration file (.CCS)
	Exit	Exit Server Script Editor. You will be prompted to save.

Menu	Menu Item	Description
	Start Test	Run the current script in test mode
	Stop Test	Stop the current script test mode
	Server Script Assistant	User script assistance interface
	Deploy Server Script	Writes (.vbs) file path to ClearView-SCADA project configuration
	Remove Server Script	Removes (.vbs) file path from ClearView-SCADA project configuration
	Help	Help notes (Update Rate)

Server Script Assistant

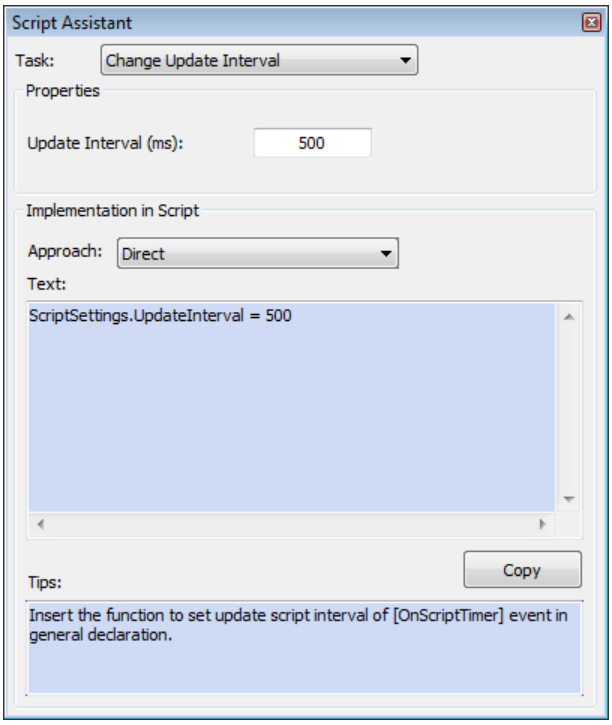


Figure #10.5A

Server Logic

Server Logic is convenient way to create automatic recipe switching based on user configurable steps. The logic is executed by ClearView-SCADA Server and runs as a process in the background regardless of a user logging into the computer similar to server-side scripting.

The logic is created in the ClearView-SCADA Client using the Server Logic editor. The editor can be accessed via the Application Explorer in the Scripting folder.

Double-clicking **Server Logic** will launch the Server Scripting Editor.

After you have created logic and saved the logic project, you can exit the Server Logic Editor and return to the ClearView-SCADA Client.

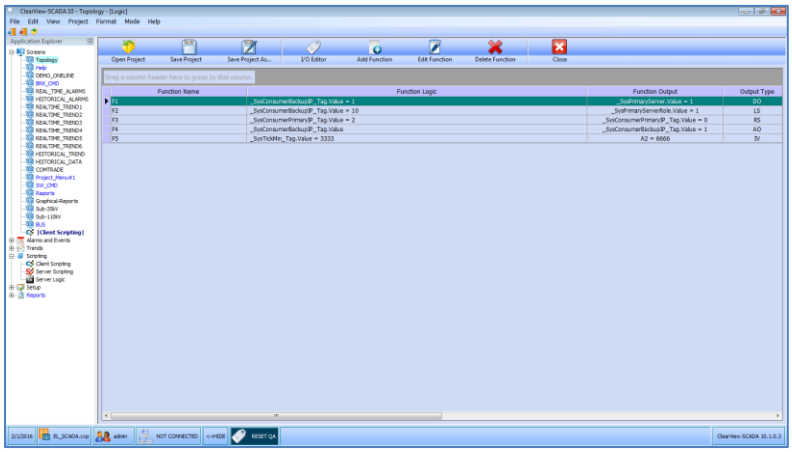


Figure #10.1A

Server Logic Editor Toolbar

Figure #10.1B

Toolbar Item	Description
Open Project	Opens saved Server Logic file (.vbs)
Save Project	Saves Server Logic file (.vbs)
Save Project As…	Saves Server Logic file in different location (.vbs)
I/O Editor	Opens I/O Editor Interface
Add Function	Opens Function Editor Interface (Add a new function)
Edit Function	Opens Function Editor Interface (modifies existing function)
Delete Function	Deletes existing function
Close	Closes the Server Logic Editor

I/O Editor

Adds Project I/O which would be used later to create project functions

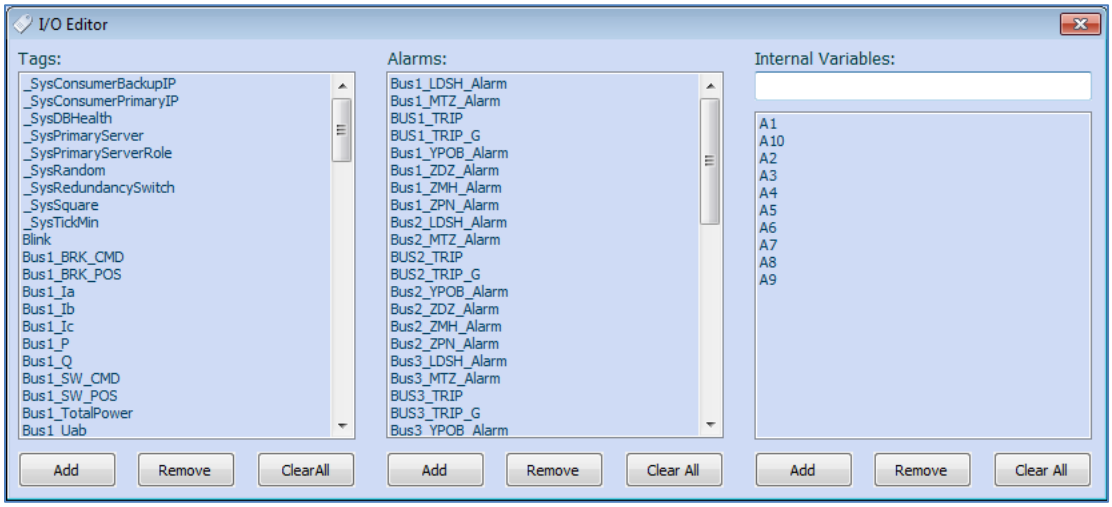


Figure #10.1C

Function Logic Editor

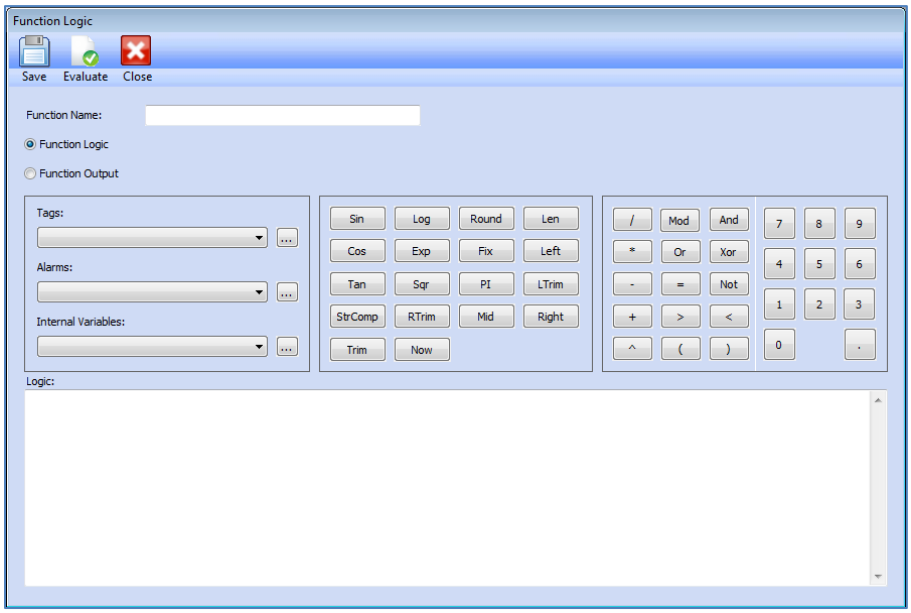


Figure #10.1D

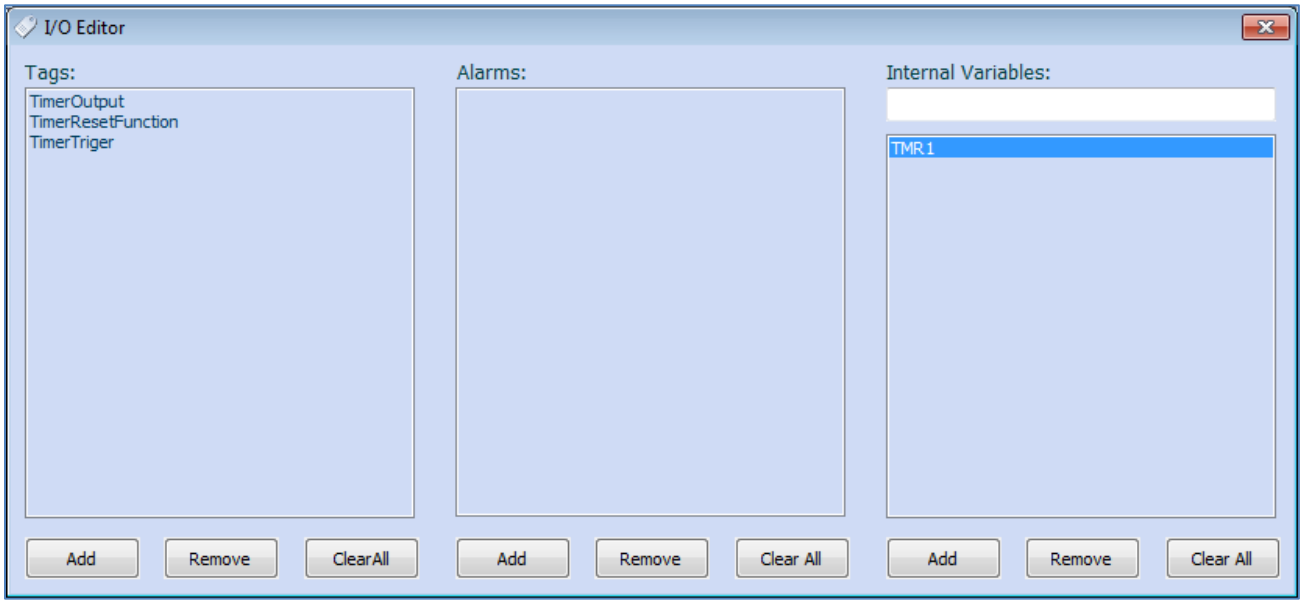
Function Logic Editor Toolbar

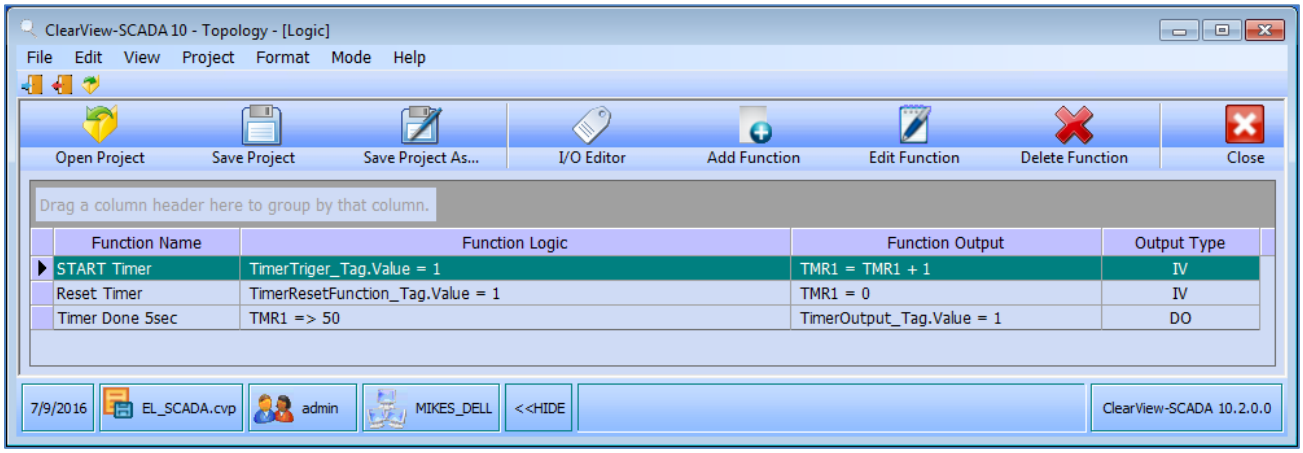
Figure #10.1E

Toolbar Item	Description
Save	Saves the current function
Evaluate	Evaluates the current function
Close	Exits the Function Logic Editor interface

Example: Create Logic Timer

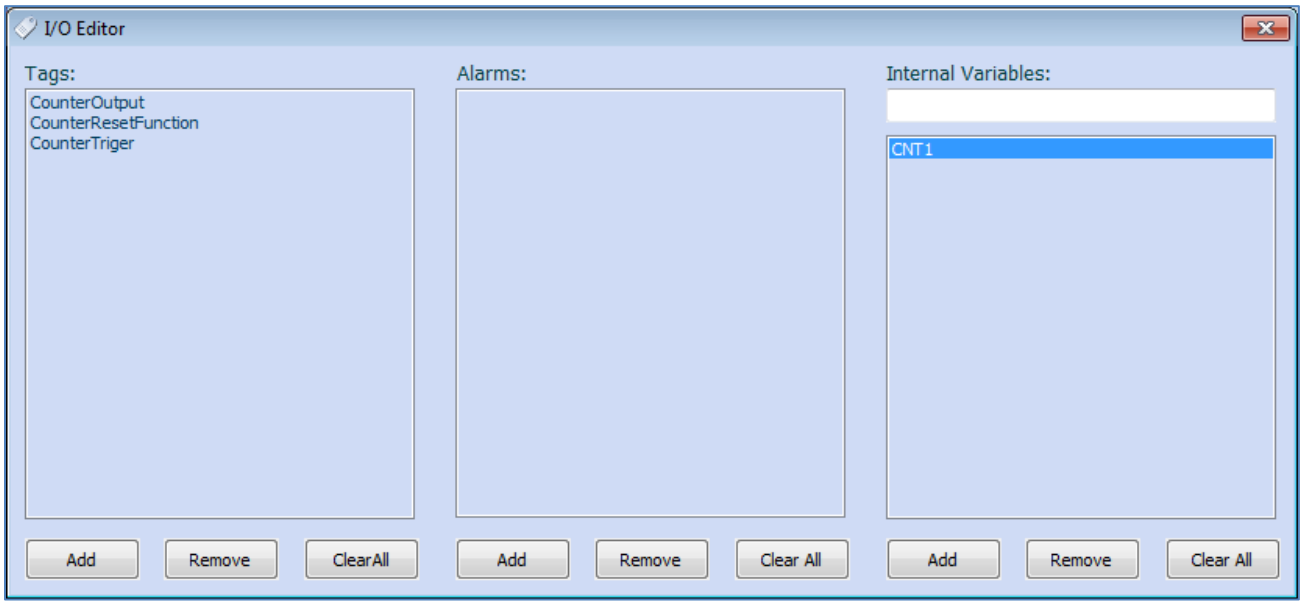
The timer pulse is 100ms. The current example illustrates 5 second timer done (TMR1 = 50).

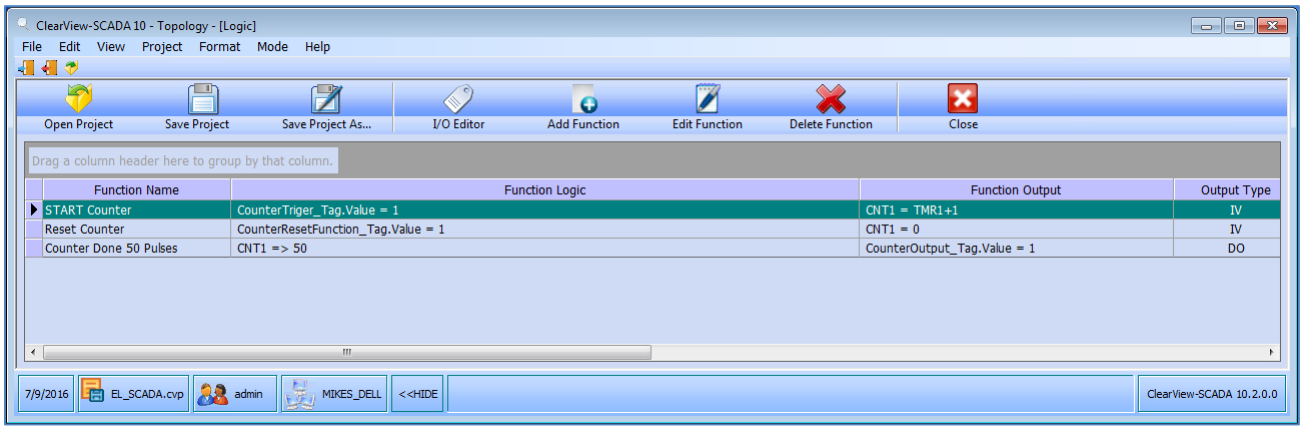




Example: Create Logic Counter

The current example illustrates 50 pulses for counter done (CNT1 = 50).



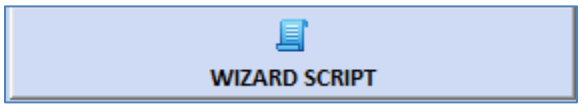


Wizard Scripting

The Wizard Script is a set of tools developed to help user to create VBScript code automatically.

To open the Wizard Script dialog box:

1. Select your ActiveX object.
2. Click **Wizard Script** button



The Wizard Script dialog box appears. Use the dialog box to select scripts if you would like to animate properties or create events.

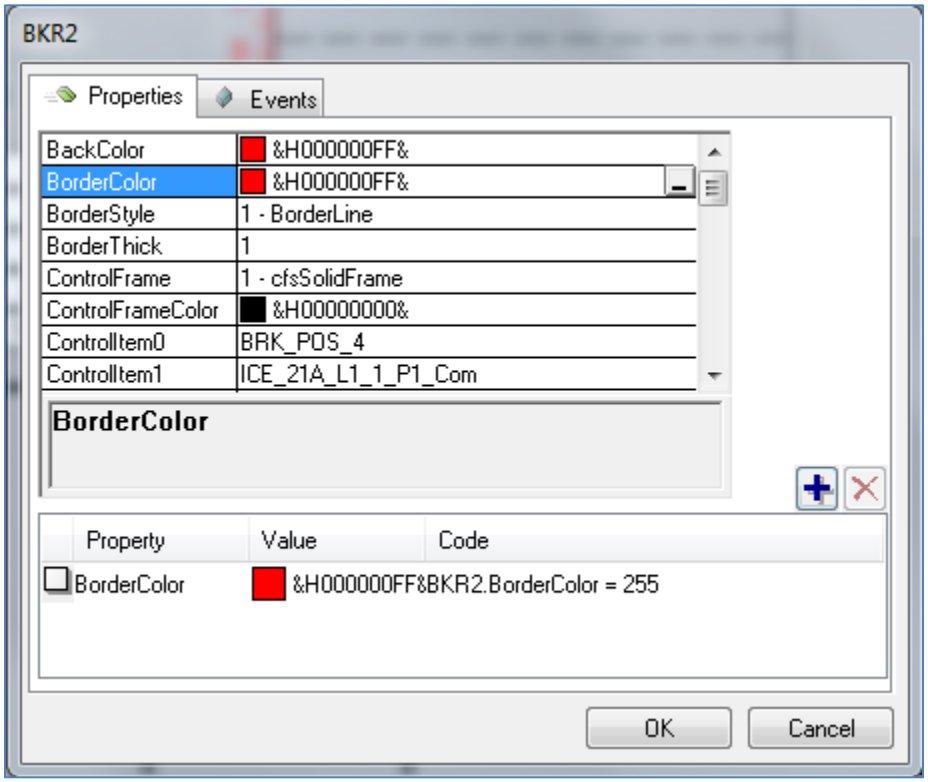


Figure #10.6

For example, if you want the BorderColor to have the property Red:

3. Select **BorderColor**, then select the value **Red**.

4. Click to add the property.

Note: To delete a property from the list you will need to select the property and click icon

If you want to script the property:

1. Double-click the new BorderColor property in the property list to open the script editor. The script editor has the most commonly used VBScript function and procedure for your object animation.

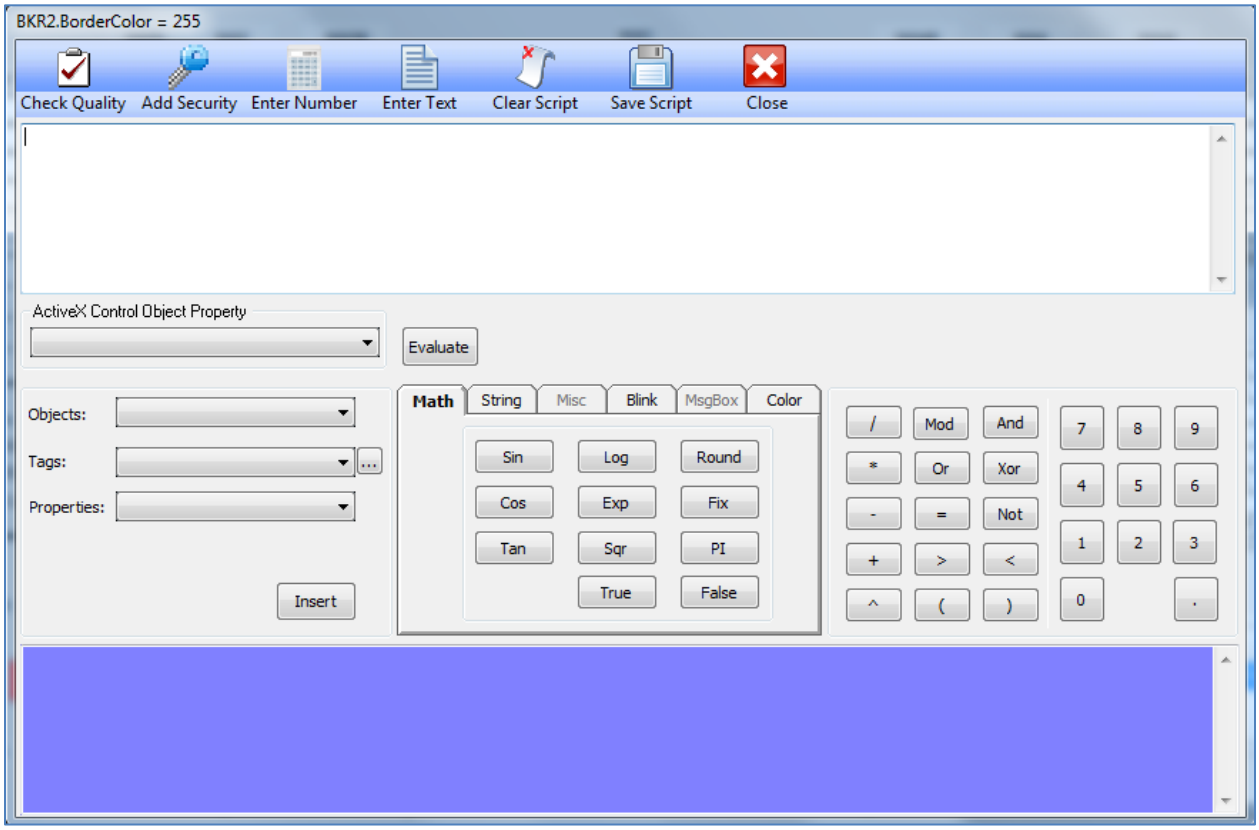


Figure #10.7

2. Verify your script by clicking **Evaluate**, then click **OK**.

Note that you can select an ActiveX Control Object Property by using “ActiveX Control Object Property” dropdown list and check Tags quality by pressing “Check Quality”

3. Select the **Events** tab in the Wizard Script if you want to modify your events script.

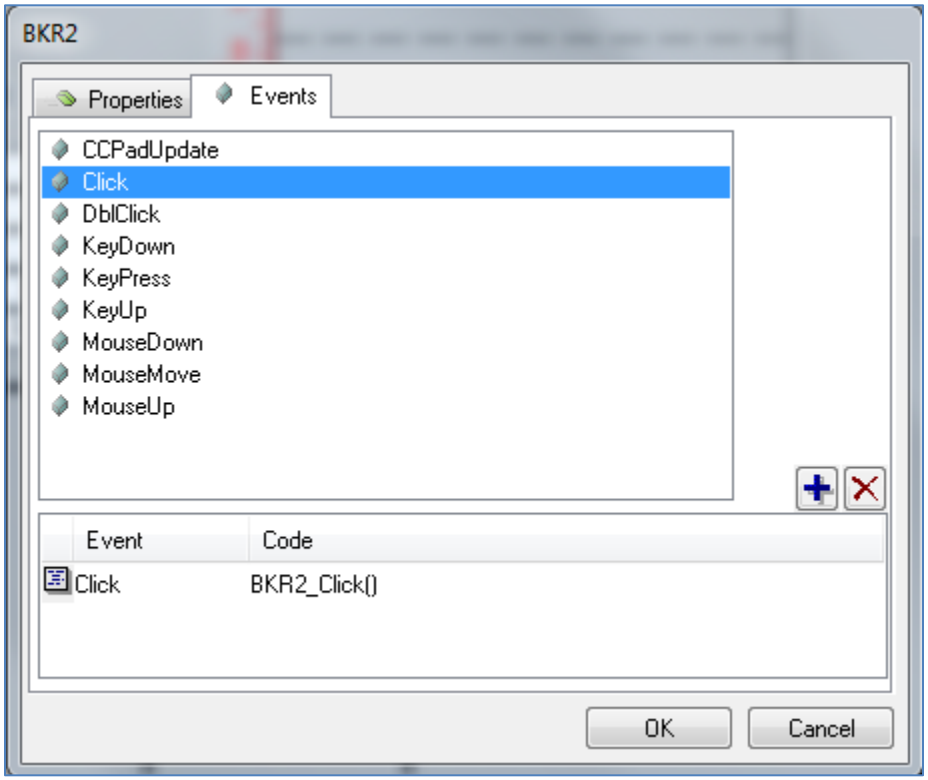


Figure #10.8

4. Select the event you want to modify and click  .

Note: To delete event from the selected list, select the event and click  .

5. Double-click the event to open the script editor.
6. The script editor has the most commonly used VBScript function and procedure for your object event creation
7. Verify your script by clicking “Evaluate” button prior to clicking “OK” button

Client Scripting

Starting from version 7.1 ClearView supports Client Scripting that provides a flexible way to create global client project scripting.

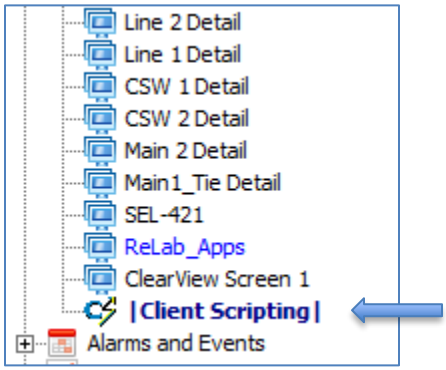


Figure #10.9

To open Client Scripting interface, double click “Client Scripting” on ClearView Application Explorer. Client Scripting configuration window will appear (Figure #10.10)

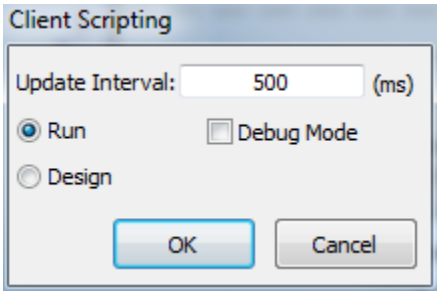


Figure #10.10

Update Interval - specifies Pad update interval in milliseconds

Run – starts script execution

Design – script design

Debug Mode – script debugging ON/OFF

OK – saves settings

Cancel – cancels operation

Clicking OK button with Design mode “ON” will open a ClearView script editor (Figure 10.11).

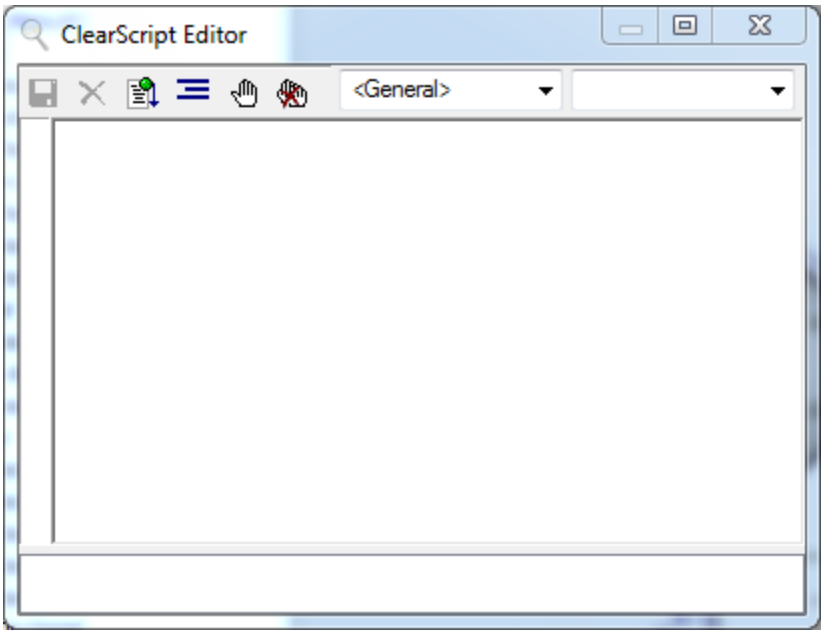


Figure #10.11

SECTION 11

ClearView Server

The ClearView server is responsible for distributing information to all clients. This is accomplished via socket connections using the TCP/IP protocol. The server maintains tag values and alarm states. The server also connects with an OPC Server to retrieve OPC tag values. An OPC server is responsible for retrieving data from a PLC, relay, or other device.

Setup

The administrator must specify server settings when a new project is being created. The settings relevant to the Server can be found under the Server and Database tabs in the Project Settings dialog box. The settings are then saved to the project file (*.cvp). The server will prompt you for the (*.cvp) file to retrieve settings upon startup.

Connecting to Clients

Clients will constantly attempt to connect to the server. A connection will be established once the server has been started.

The (*.CVP) file includes the following settings that are used by the server:

Database Tab

- Data Provider
- Data Source
- User Name
- Password

Server Tab

- Enable Redundancy
- Enable Server Scripting
- OPC Tags Update Interval

Please see Project Settings for more information.

Choosing a database type

This section discusses some limitations to consider when choosing which database server to use.

- Microsoft Access Database
- Microsoft SQL Server (require to install SQLDMO 64-bit or 32-bit on SQL Servr 2012 and higher)
- Oracle Server

Changing Server Settings

All changes to the ClearView Server settings are done under ClearView Client Project settings. If the settings are changed, the server will need to be stopped and restarted.

Changing Database

Initially, you may have developed the project using a less capable database, for example, MS Access. However, as the system grows you may want to upgrade to a full database management system. There are several possible scenarios for doing this.

Scenario	Instructions
MS Access to MS Access	1. MS Access database consist of one file. The file is of type (*.mdb). Copy the (*.mdb) file to a new location and rename it. 2. Point the data source to a new (*.mdb) file. Retrieve name of data source from Project Settings on the Database tab. 3. In the Window's Control Panel, select **Administrative Tools** folder, and then click the Data Sources (ODBC) icon. 4. Find the data source under the System DSN tab. 5. Click **Configure**. 6. Click **Select** and locate new (*.mdb) file. 7. Restart ClearView.
MS Access to *SQL SERVER*	SQL SERVER is unlike MS Access because it is a database management system and consists of more than one file. 1. Move from MS Access to SQL SERVER by using the **Backup/Restore** tool. See Backup/Restore for more information.
Any type <-> CSV for Tags and Alarms	1. Start **ClearView Server..** 2. Navigate to Tags Database or Alarm Database. 3. Click **Import/Export**.

Starting the ClearView Server

First-Time Start

Note: You must create a ClearView Project file (*.cvp) before starting the server. The project file is built using ClearView Client software: ClearView.exe.

1. On Window's **Start** menu, select **Programs\Clear Controls\ClearView\ClearView Server**. A message box will prompt you to select a project.
2. Press **OK**.
3. On the Open dialog box, select and open your ClearView Project file (***.cvp**). The ClearView Server dialog box, shown below, will appear. The ClearView Server icon will also appear in the system tray.
4. On the ClearView Server dialog box, click **Autoload Settings** to always open the same project file each time the server is restarted.
5. Hide the Server dialog box without shutting down the server by pressing the **Cancel** button.

Figure #11.1

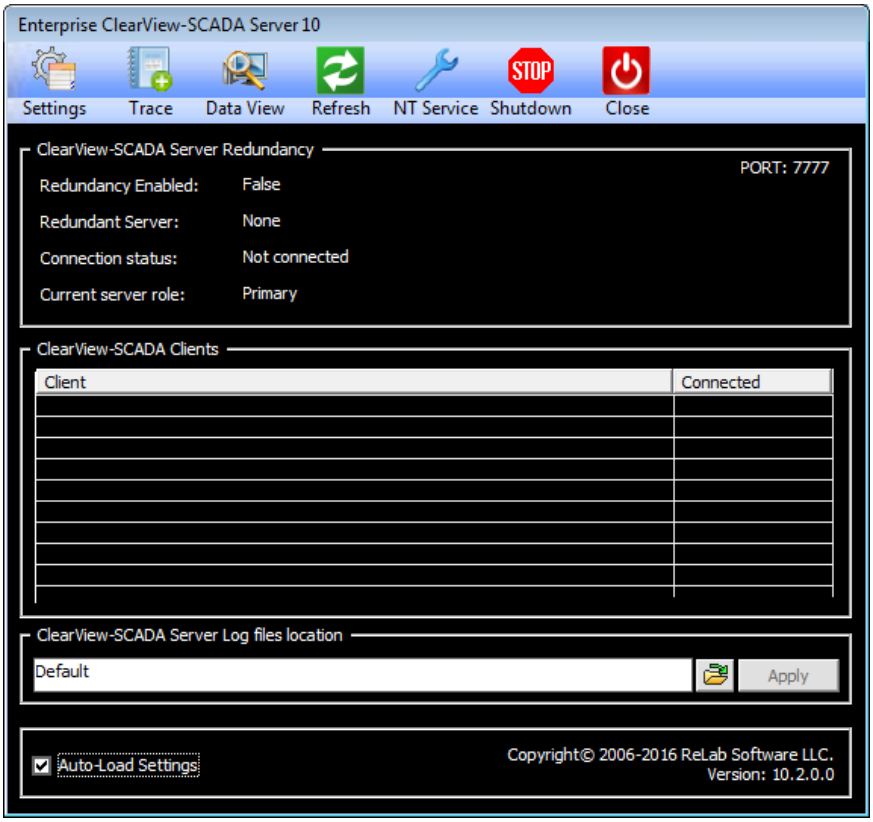


Figure #11.2

Note: To redisplay the ClearView Server dialog box, double-click the server icon in the system tray.

Server Redundancy

Server redundancy requires two servers to be running on the same network. In the case of a primary server failure, the client will connect to the Redundant Server.

To setup **Server Redundancy,** check the option in Project Settings on the Server tab.

After loading the project file, if redundancy is enabled, the server will automatically connect to the redundant server.

If redundancy is not enabled, then return to Project Settings and follow the procedure to enable redundancy.

If the connection is successful, the Connection Status changes from “not connected” to “connected”.

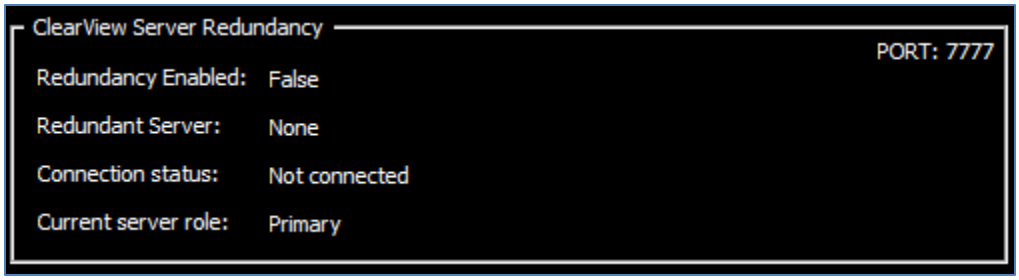


Figure #11.3

The Current Server Role must also be selected. Options include:

- **Primary**: The active server. Accepts tag updates and sends updates to clients
- **Backup**: The server is not active but is ready to assume the Primary role if needed.

Note: The Backup Server does not accept tag updates and does not send updates to clients.

Client List

The ClearView Clients section, directly below the Redundancy section, displays a list of clients connected to the server. The Refresh button will update the information about server redundancy and client connections.

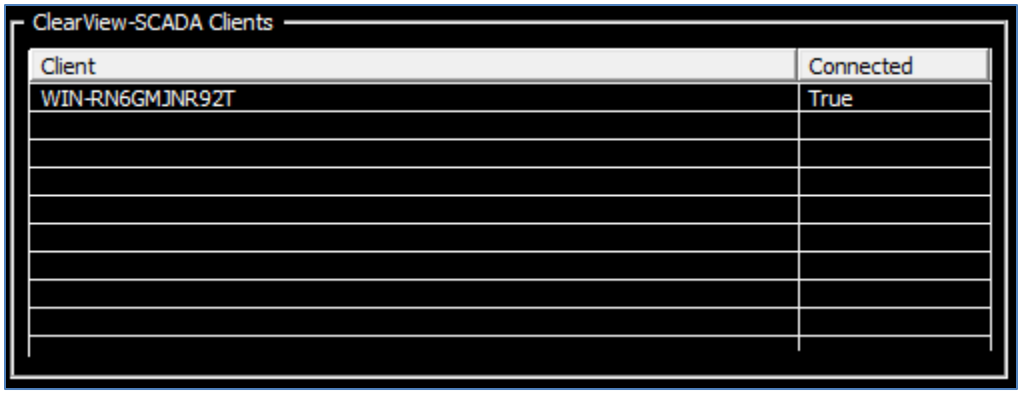


Figure #11.4

Server Settings

Click **Settings** 

to display the server configuration settings. To make changes to the server settings, configure the ClearView Client settings, save the project file, restart the server, and select the saved project file.

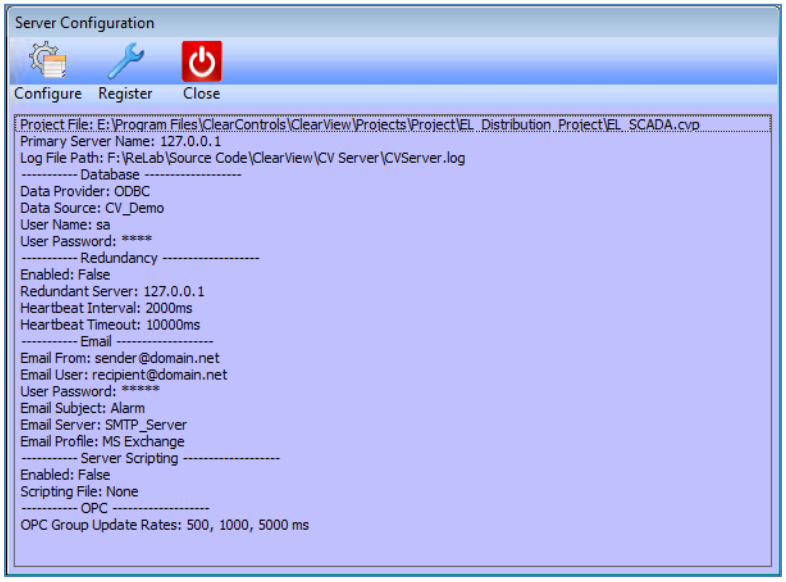


Figure #11.5

Trace

Click **Trace** to display a list of server events. This screen displays a list of all events that have occurred on the server.

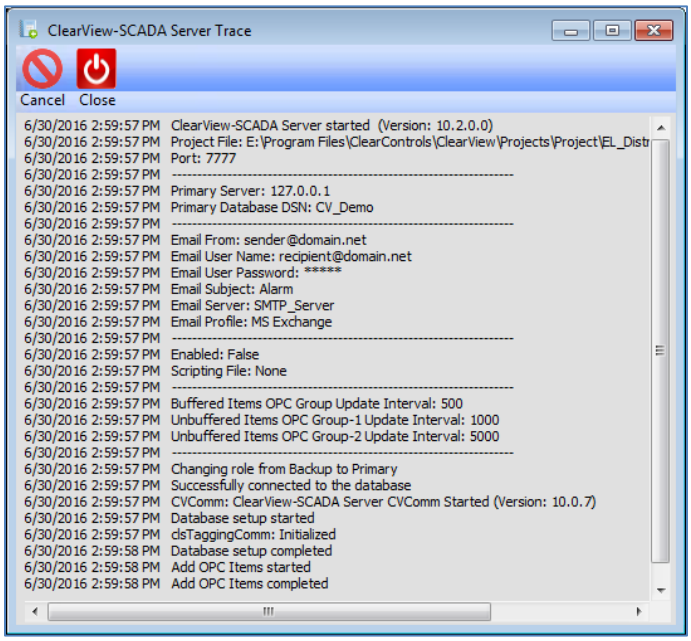


Figure #11.6

Exit button closes the Server Trace form.

Close button closes the form and clears the trace.

Data View

Click **Data** to display Alarm & Tags information. The Data View screen allows the user to view two displays, a list of tags and their current values and a list of active alarms.

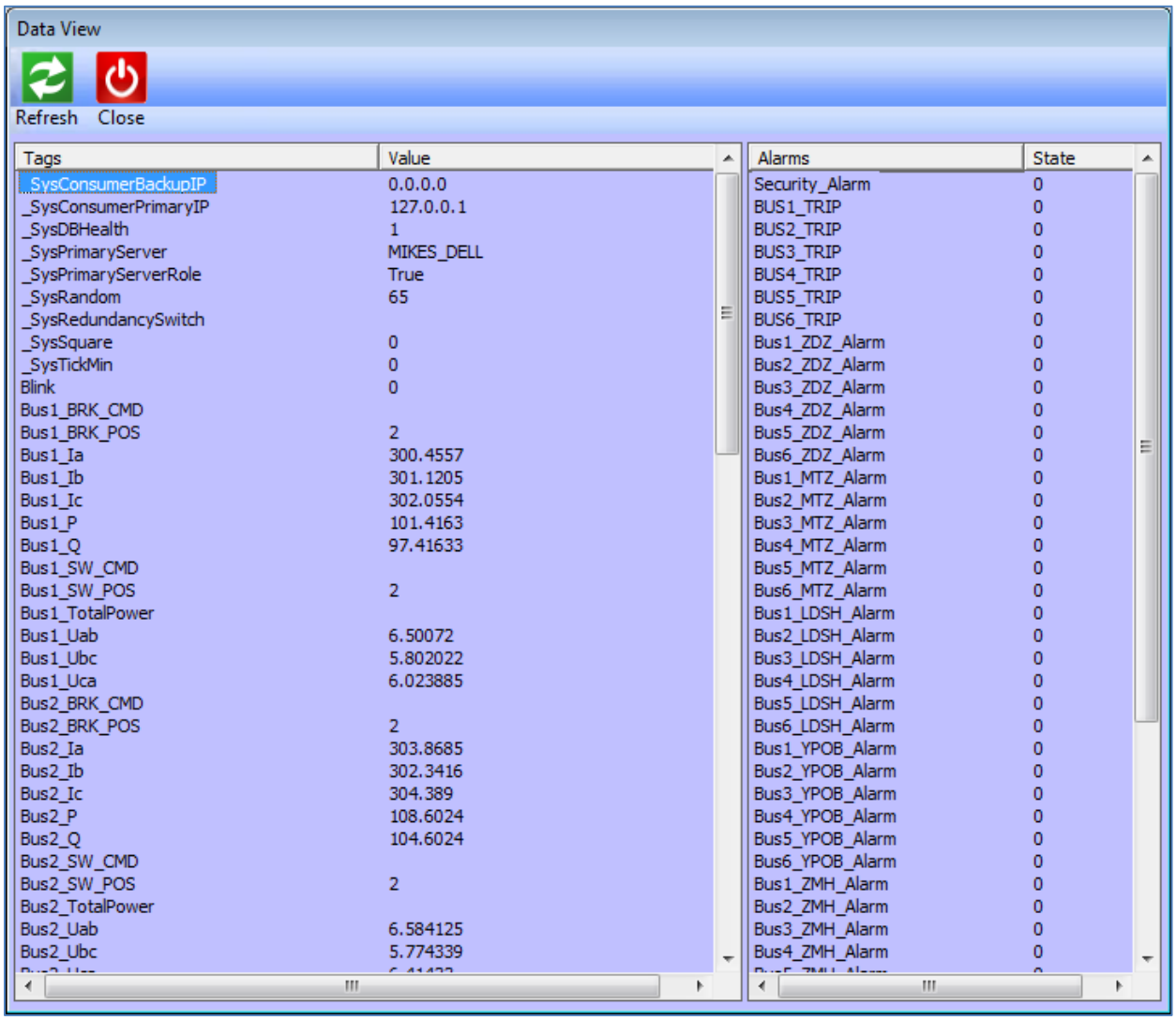


Figure #11.7

NT Service

How to Start ClearView Server as an NT Service

The following procedure describes how to start ClearView Server as an NT service.

1. Click **NT Service** to configure ClearView Server.

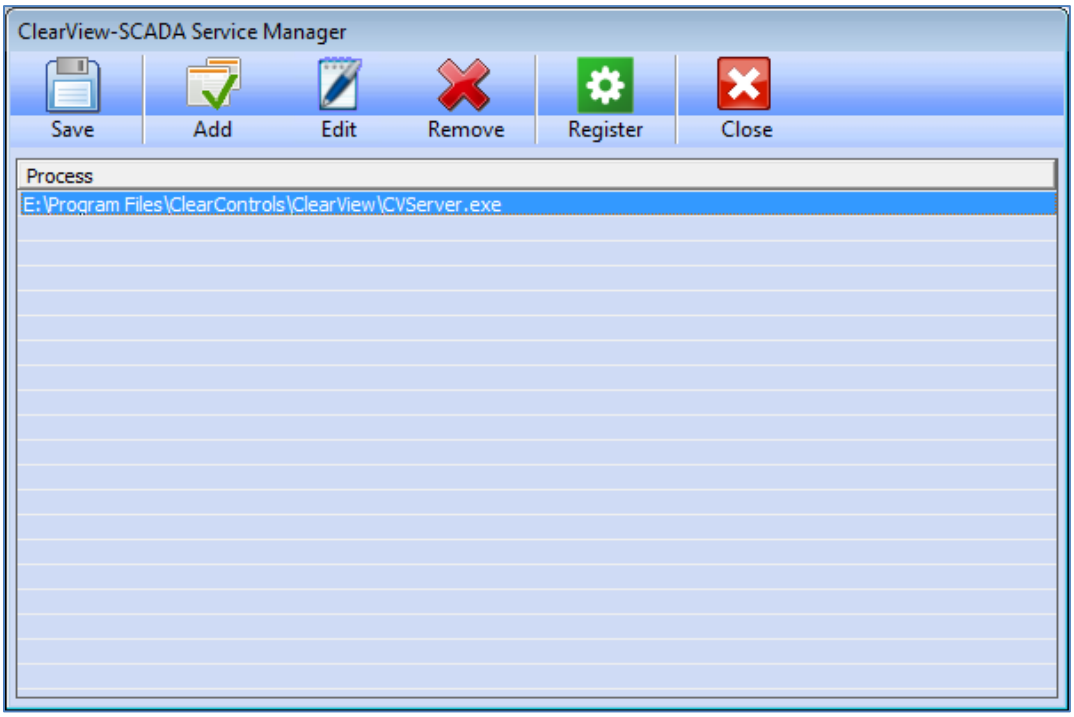


Figure #11.8

2. Click **Add**.

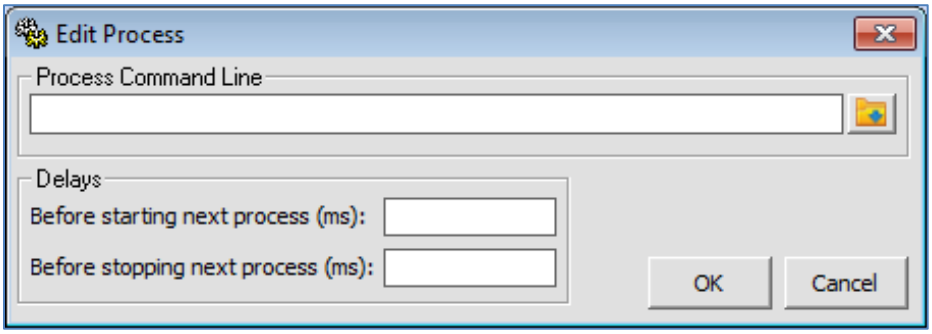


Figure #11.9

3. Click **Browse** to locate and ClearView Server application file: **CVServer.exe**
4. Enter a delay time, in milliseconds, before starting next process. Example: 30000 milliseconds (30 seconds).
5. Enter a delay time, in milliseconds, after stopping next process. Example: 30000 milliseconds (30 seconds).
6. Click **OK**.
7. Click **Register** to register ClearView Server as a service.
8. Click **Save**.
9. Click **Exit.**
10. You should be able to see ClearView Service in Windows Service Manager, Figure 11.10.

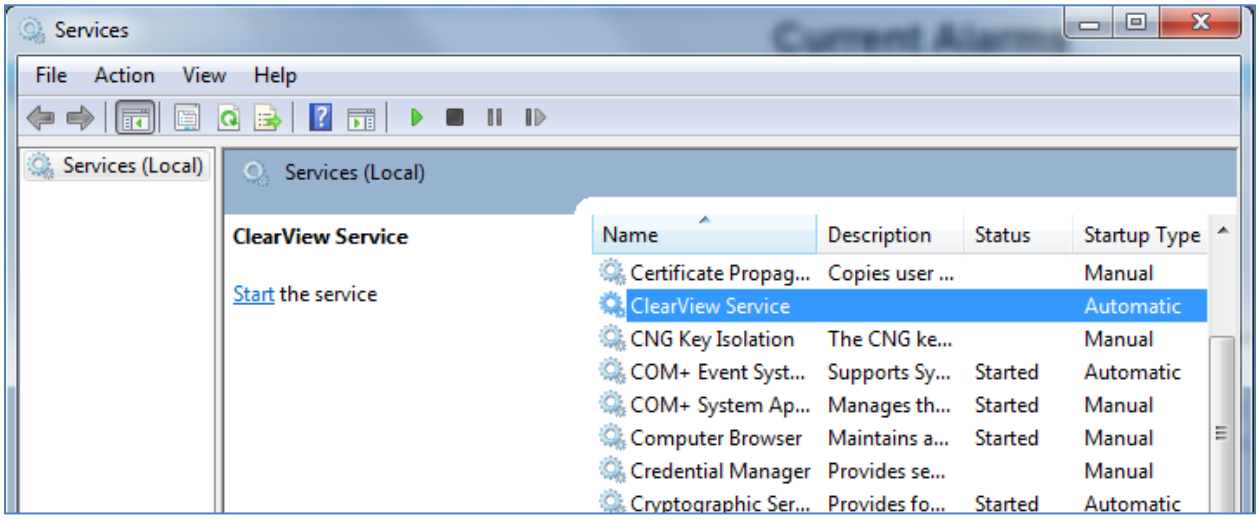


Figure #11.10

Tip: To ensure that the database server, like SQL Server, has had enough time to startup, add an additional service placed before starting CVServer. A possible simple service would be a *.bat file that synchronizes time to a server on the network. Enter a comfortable time before starting the next process, CVServer or set proper service dependencies.

Note: ClearView Server will fault on startup if the database server, like Microsoft SQL or ORACLE, has not had sufficient time to startup first. Make sure that service dependencies are configured:

- Stop "ClearView Service" service
- Open command line (CMD.exe) interface as Administrator
- Enter the following line to set dependencies: **sc config "ClearView Service" depend= "Service Name (Oracle or SQL Server)"**
- If multiple dependencies are required the "/" separator can be used (example: sc config ServiceA depend= ServiceB/ServiceC/ServiceD)

Cancel

The **Cancel** button will minimize the server window, making it appear as an icon in the system tray but not in the Taskbar.

Shutdown

The **Shutdown** button will close or end the ClearView Server session.

Server Log Files Location

By default, ClearView Server and ClearView Client logs are located in the same directory where ClearView was installed. By changing **ClearView-SCADA Server Log Files Location** path user can specify the directory where ClearView Server will write its logs.

Autoload Setting

The Autoload Settings check box lets the Server program "remember" which project was used last. When the check box is selected, the Server program will automatically open the last project file that was used. When the Server is first started the user must select a project and all the settings come from that project. If a user is working on one project for

they should use the Autoload Settings feature by checking the check box so they do not have to repeatedly select the same project.

SECTION 12

ClearView Reports

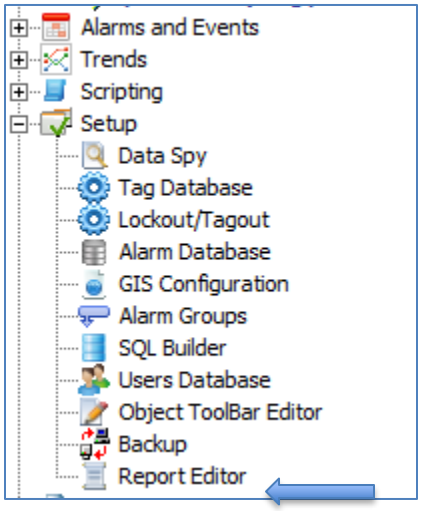


Figure #12.1

To open report editor, double click “Report Editor” on ClearView Application Explorer.

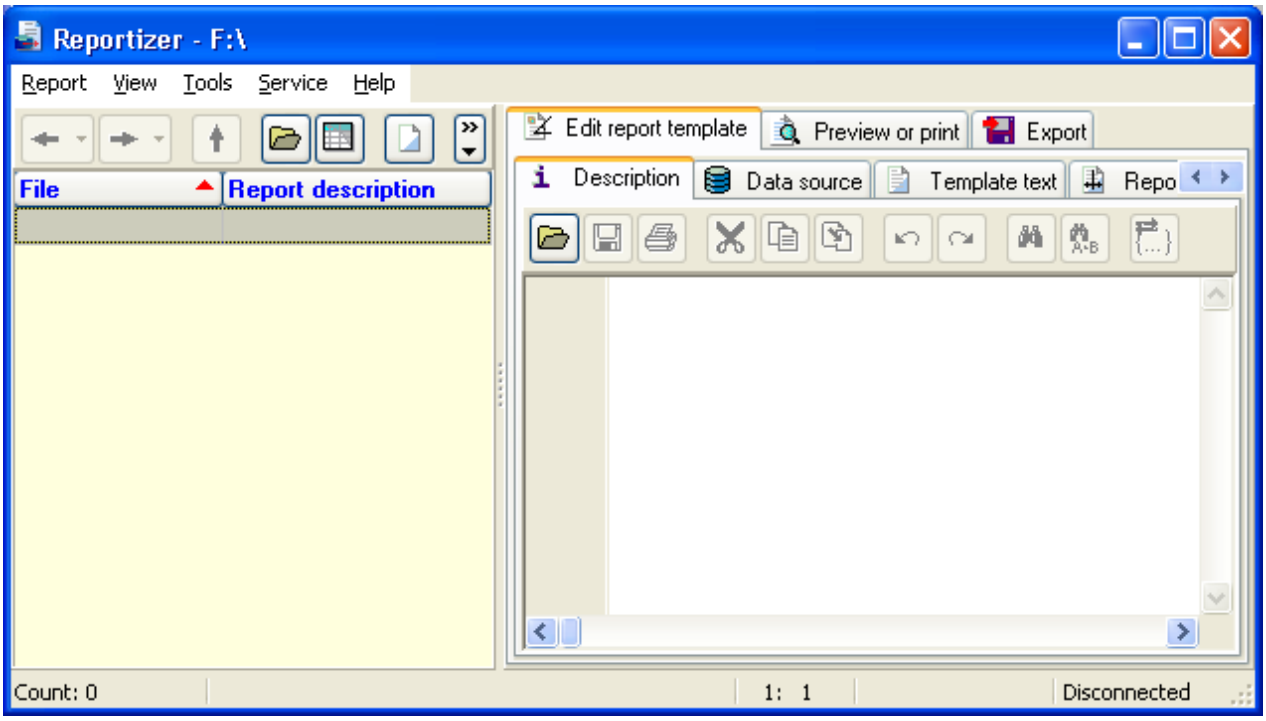


Figure #12.2

Note: Refer to “Reportizer” manual to configure your reports

Reports Interface

This chapter describes command-line interface between ClearView-SCADA and ClearView Reporting Module. ClearView "Open File" ActiveX object provides ability for a direct interface to user-configured reports by executing command-line arguments procedures.

Configuring ClearView Reports Interface

The following functions can be performed.

- Open report in Design mode
- View reports
- Print reports
- Query reports

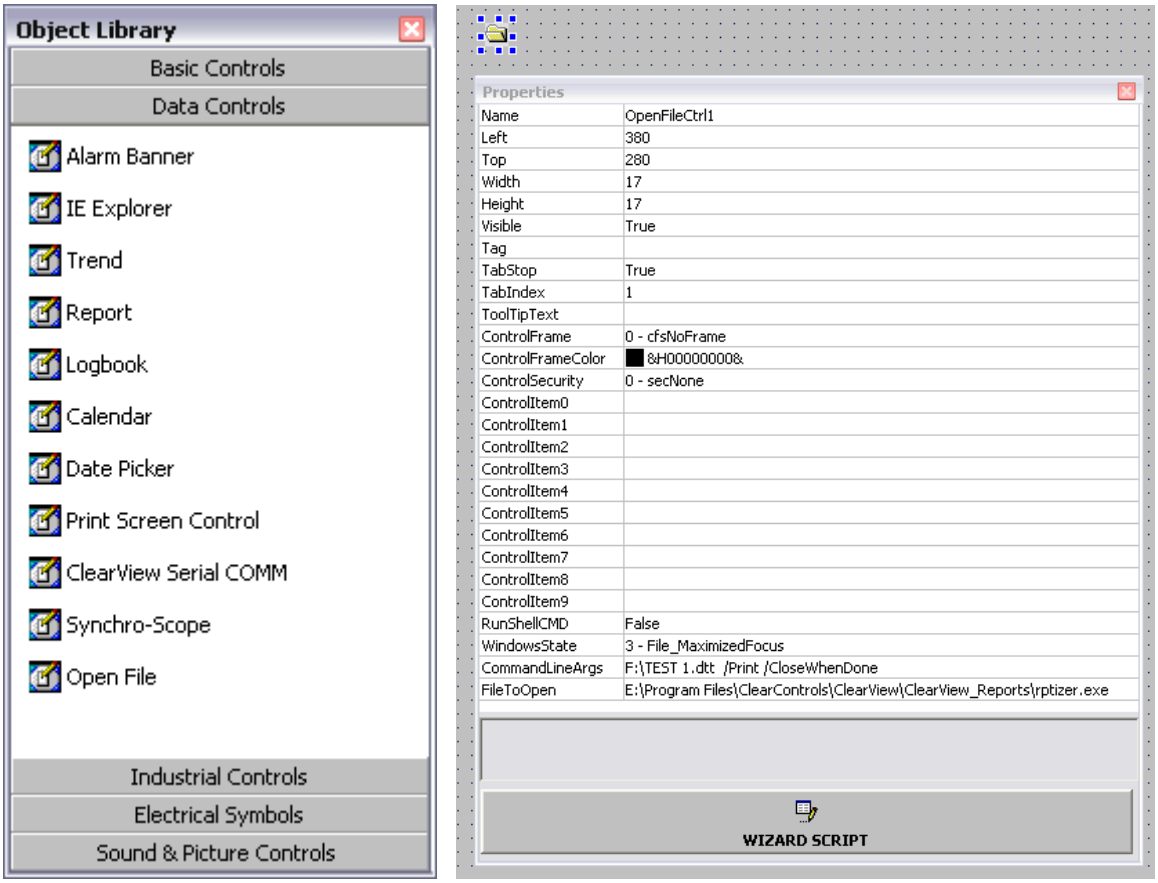


Figure #12.3

ClearView Report Interface Arguments

Argument	Description	Example
/CloseWhenDone	Specifies that the program is closed after successful executing of the specified operation (for example, report printing).	Private Sub Open_Click( ) 'Specifies report executable location (full path) OpenFileCtrl1.FileToOpen = "E:\Program Files\ClearControls\ClearView\ClearView_Reports\rptizer.exe " 'F:\TEST 1.dtt specifies report location (full path) OpenFileCtrl1.CommandLineArgs = "F:\TEST 1.dtt /Print /CloseWhenDone" 'Executes procedure OpenFileCtrl1.OpenFile End Sub
/UserName	Specifies the database user name that can access the dynamic report database (that is, if the report database is remote, the database that requires user name to access it). This switch is used with /print or /preview switches.	Private Sub Open_Click( ) 'Specifies report executable location (full path) OpenFileCtrl1.FileToOpen = "E:\Program Files\ClearControls\ClearView\ClearView_Reports\rptizer.exe " 'F:\TEST 1.dtt specifies report location (full path) OpenFileCtrl1.CommandLineArgs = "F:\TEST 1.dtt /Print /Username = Name /CloseWhenDone" 'Executes procedure OpenFileCtrl1.OpenFile End Sub
/Password	Specifies the database password to access the dynamic report database (that is, if the report database is remote, the database that requires password to access it). This switch is used with /print or /preview switches.	Private Sub Open_Click( ) 'Specifies report executable location (full path) OpenFileCtrl1.FileToOpen = "E:\Program Files\ClearControls\ClearView\ClearView_Reports\rptizer.exe " 'F:\TEST 1.dtt specifies report location (full path) OpenFileCtrl1.CommandLineArgs = "F:\TEST 1.dtt /Print /Password = Password /CloseWhenDone" 'Executes procedure OpenFileCtrl1.OpenFile

Argument	Description	Example
		End Sub
/PrinterName	Specifies the printer that prints the report. This switch overrides the printer settings, saved in the report. This switch is used with /print or /preview switches.	Private Sub Open_Click( ) 'Specifies report executable location (full path) OpenFileCtrl1.FileToOpen = "E:\Program Files\ClearControls\ClearView\ClearView_Reports\rptizer.exe " 'F:\TEST 1.dtt specifies report location (full path) OpenFileCtrl1.CommandLineArgs = "F:\TEST 1.dtt /Print /PrinterName = My Printer /CloseWhenDone" 'Executes procedure OpenFileCtrl1.OpenFile End Sub
/PaperSize	Specifies the printer paper size (for example, Letter, Legal, etc.).	Private Sub Open_Click( ) 'Specifies report executable location (full path) OpenFileCtrl1.FileToOpen = "E:\Program Files\ClearControls\ClearView\ClearView_Reports\rptizer.exe " 'F:\TEST 1.dtt specifies report location (full path) OpenFileCtrl1.CommandLineArgs = "F:\TEST 1.dtt /Print /PaperSize = Legal /CloseWhenDone" 'Executes procedure OpenFileCtrl1.OpenFile End Sub
/PaperSource	Specifies the paper source for a printer that supports several paper sources.	Private Sub Open_Click( ) 'Specifies report executable location (full path) OpenFileCtrl1.FileToOpen = "E:\Program Files\ClearControls\ClearView\ClearView_Reports\rptizer.exe " 'F:\TEST 1.dtt specifies report location (full path) OpenFileCtrl1.CommandLineArgs = "F:\TEST 1.dtt /Print /PaperSource = Paper Source /CloseWhenDone" 'Executes procedure OpenFileCtrl1.OpenFile

Argument	Description	Example
		End Sub
/DuplexMode	Specifies the duplex mode for a printer that supports duplex printing.	Private Sub Open_Click( ) 'Specifies report executable location (full path) OpenFileCtrl1.FileToOpen = "E:\Program Files\ClearControls\ClearView\ClearView_Reports\rptizer.exe" 'F:\TEST 1.dtt specifies report location (full path) OpenFileCtrl1.CommandLineArgs = "F:\TEST 1.dtt /Print /DuplexMode = Duplex Mode /CloseWhenDone" 'Executes procedure OpenFileCtrl1.OpenFile End Sub
/Preview	Specifies the report must be opened in preview mode.	Private Sub Open_Click( ) 'Specifies report executable location (full path) OpenFileCtrl1.FileToOpen = "E:\Program Files\ClearControls\ClearView\ClearView_Reports\rptizer.exe" 'F:\TEST 1.dtt specifies report location (full path) OpenFileCtrl1.CommandLineArgs = "F:\TEST 1.dtt /Preview" 'Executes procedure OpenFileCtrl1.OpenFile End Sub
/Print	Specifies the report must be printed.	Private Sub Open_Click( ) 'Specifies report executable location (full path) OpenFileCtrl1.FileToOpen = "E:\Program Files\ClearControls\ClearView\ClearView_Reports\rptizer.exe" 'F:\TEST 1.dtt specifies report location (full path) OpenFileCtrl1.CommandLineArgs = "F:\TEST 1.dtt /Print /CloseWhenDone" 'Executes procedure OpenFileCtrl1.OpenFile End Sub

Argument	Description	Example
/Stretchprint	Specifies the report must be printed in stretch print mode.	Private Sub Open_Click( ) 'Specifies report executable location (full path) OpenFileCtrl1.FileToOpen = "E:\Program Files\ClearControls\ClearView\ClearView_Reports\rptizer.exe" 'F:\TEST 1.dtt specifies report location (full path) OpenFileCtrl1.CommandLineArgs = "F:\TEST 1.dtt /Print /Stretchprint /CloseWhenDone" 'Executes procedure OpenFileCtrl1.OpenFile End Sub
/Template	Specifies the report template that must be opened in Design mode (dynamic reports only).	Private Sub Open_Click( ) 'Specifies report executable location (full path) OpenFileCtrl1.FileToOpen = "E:\Program Files\ClearControls\ClearView\ClearView_Reports\rptizer.exe" 'F:\TEST 1.dtt specifies report location (full path) OpenFileCtrl1.CommandLineArgs = "F:\TEST 1.dtt /template" 'Executes procedure OpenFileCtrl1.OpenFile End Sub
/ds	Specifies the report template that must be opened in data source editing mode (dynamic reports only).	Private Sub Open_Click( ) 'Specifies report executable location (full path) OpenFileCtrl1.FileToOpen = "E:\Program Files\ClearControls\ClearView\ClearView_Reports\rptizer.exe" 'F:\TEST 1.dtt specifies report location (full path) OpenFileCtrl1.CommandLineArgs = "F:\TEST 1.dtt /ds" 'Executes procedure OpenFileCtrl1.OpenFile End Sub
/queryparam_%paramname%	For dynamic reports, where the report dataset is a	Private Sub Open_Click( )

Argument	Description	Example
	parameterized query. Use this switch to define type (%paramtype% specificator) and value (%paramvalue% delimiter) of the parameter specified by name %paramname% (the colon must be omitted here). Available types: Boolean, Currency, Date, DateTime, Float, Integer, SmallInt, String, Time, Word. Type and value of the parameter must be enclosed in square brackets. To pass null value, text of the %paramvalue% delimiter should be [NULL] (in upper case). If the switch contains spaces, the switch should be enclosed in quotes. You may specify as many such switches as you need.	'Specifies report executable location (full path) OpenFileCtrl1.FileToOpen = "E:\Program Files\ClearControls\ClearView\ClearView_Reports\rptizer.exe" 'F:\TEST 1.dtt specifies report location (full path) OpenFileCtrl1.CommandLineArgs = "F:\TEST.dtt /Print /queryparam_%paramname%=%paramtype%%paramvalue%" 'Executes procedure OpenFileCtrl1.OpenFile End Sub
/StaticPassword	Specifies password for password protected static report.	Private Sub Open_Click( ) 'Specifies report executable location (full path) OpenFileCtrl1.FileToOpen = "E:\Program Files\ClearControls\ClearView\ClearView_Reports\rptizer.exe" 'F:\TEST 1.dtt specifies report location (full path) OpenFileCtrl1.CommandLineArgs = "F:\TEST.dtt /Print /StaticPassword = Password" 'Executes procedure OpenFileCtrl1.OpenFile End Sub

SECTION 13

ClearView-SCADA Geographic Information System (GIS)

Registering GIS

1. Open ClearView-SCADA Client.
2. Navigate to **Menu > Help > Activate GIS**.

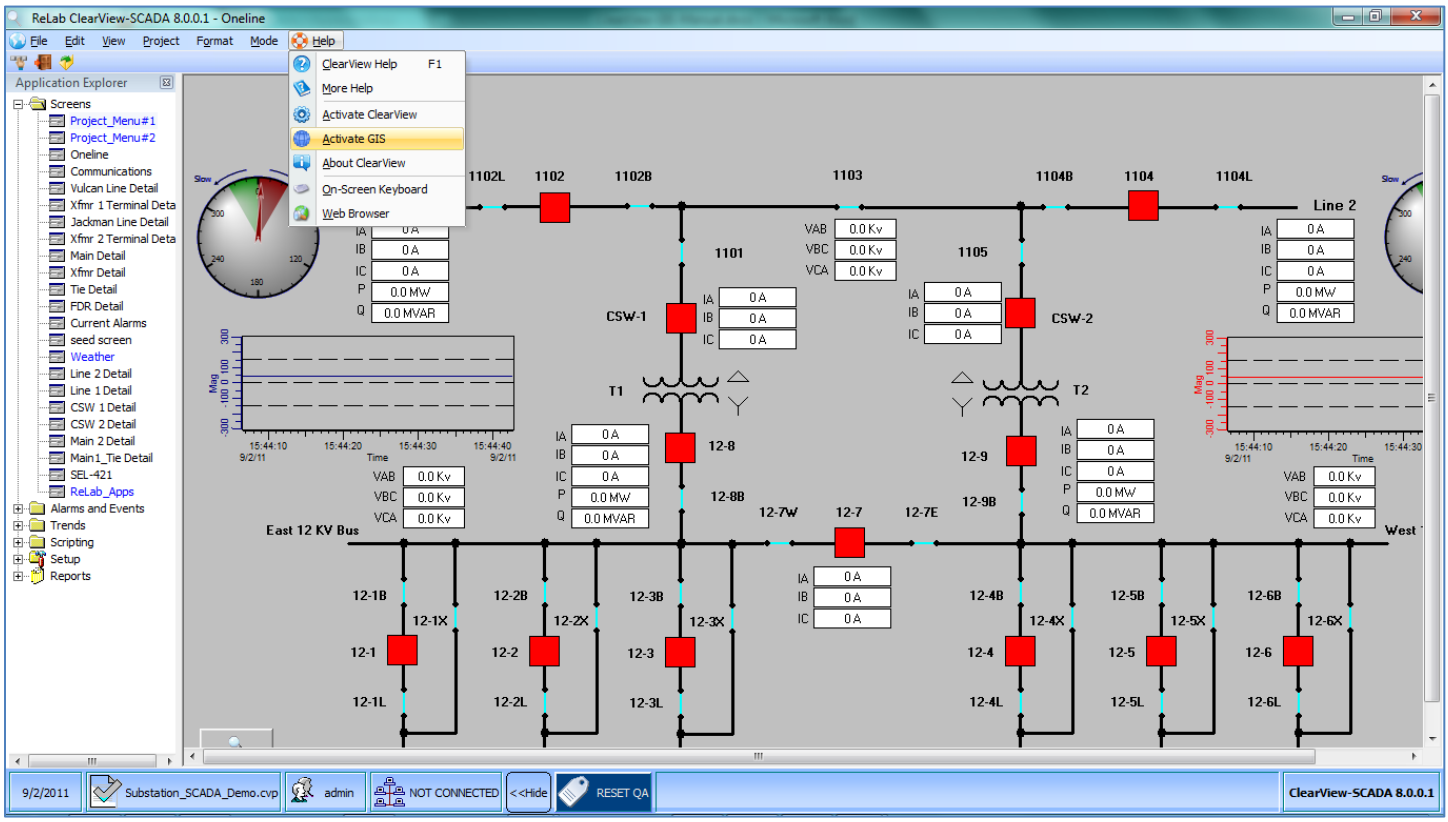


Figure #13.1

3. Contact ReLab Software and provide the **Customer ID** number displayed in the dialog box.

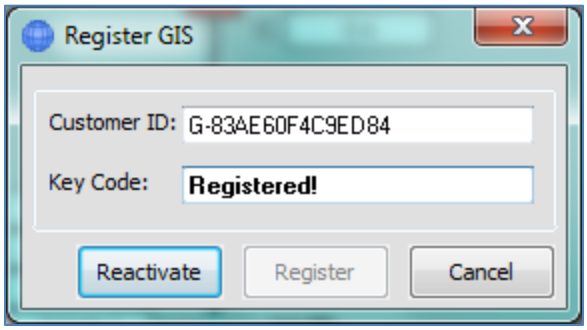


Figure #13.2

4. Enter the **Key Code** provided by ReLab Software.
5. Click **Register** button.

6. Restart ClearView-SCADA.

Enabling GIS

To enable GIS for a specific project:

1. Navigate to **Menu > File > Project Settings**.
2. Select **Enable GIS** check box
3. **Save** the project.
4. Restart Clear View – SCADA client.

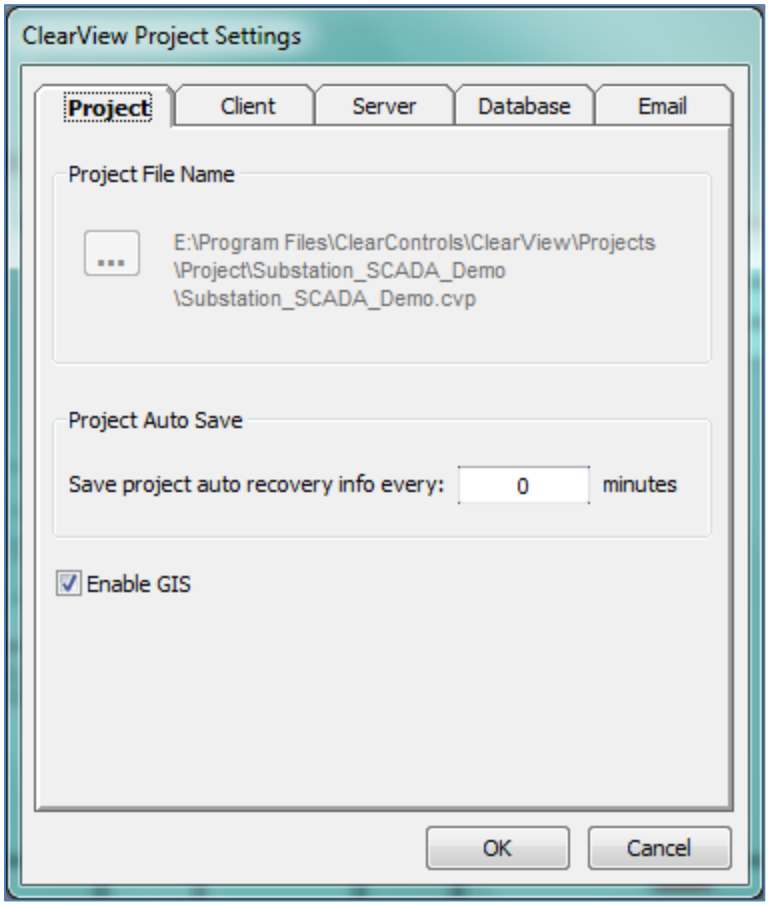


Figure #13.3

Configuring GIS

Opening the Configuration Interface

In **Application Explorer,** double-click **GIS Configuration** under the **Setup** folder. You will be asked if you want to create a new configuration if there is no such in the project yet. After answering "Yes" you need to provide the location for the first map. Answering "No" will stop GIS Configuration.

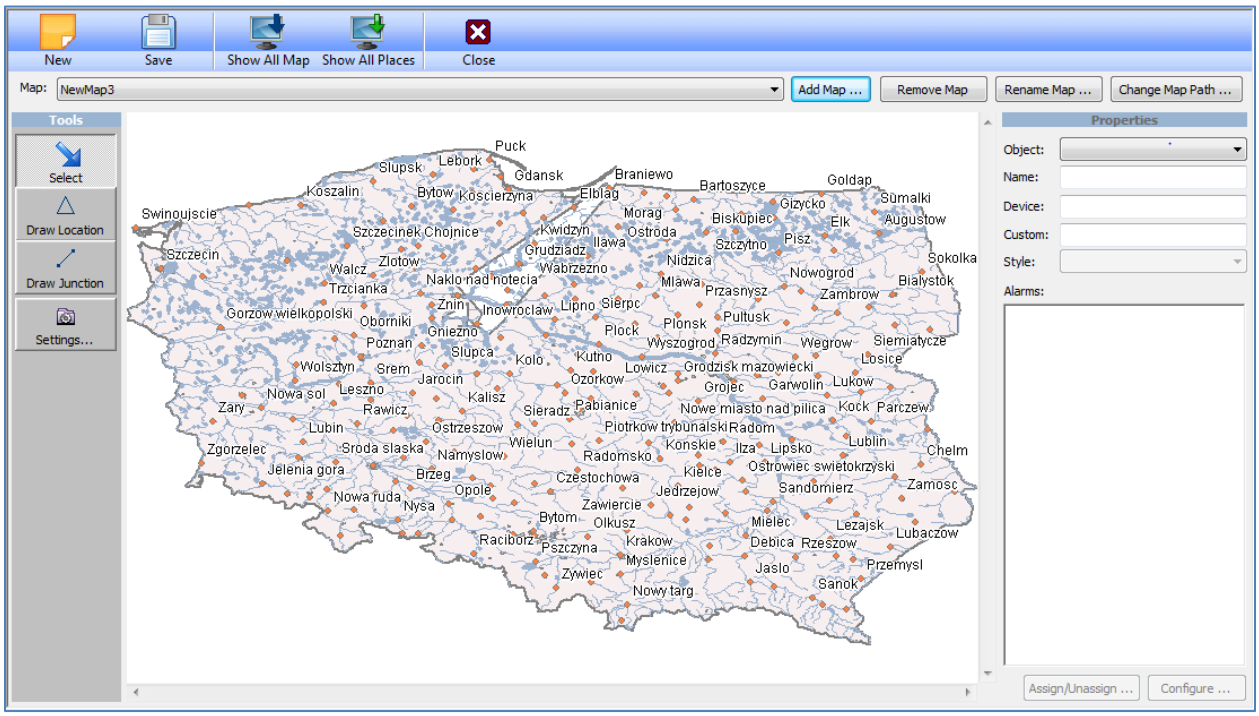


Figure #13.4

Adding and Removing Maps

To add a map, click **Add Map**, select the file, and click **Open**.

GIS support the following map file types:

- Bitmap (*.bmp)
- Microstation (*.dgn)
- AutoCad Export Format (*.dxf)
- Joint Photographic Exports Group (*.jpeg)
- Portable Network Graphics (*.png)
- Shape (*.shp)
- Tagged Image File (*.tif and *.tiff)
- Tatuk GIS Project (*.ttkjp)

ClearView installation provides sample maps. The maps by default are installed to **C:\Program Files\ClearControls\ClearView\Projects\GIS_MAPS** folder.

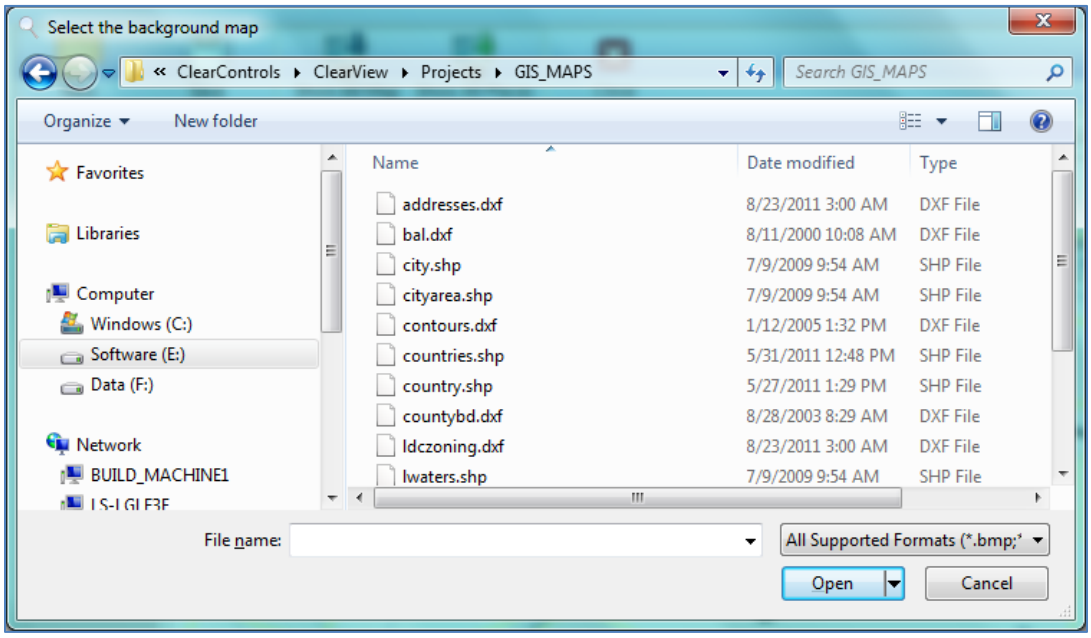


Figure #13.5

To delete a map: select the map and then click **Remove Map**.

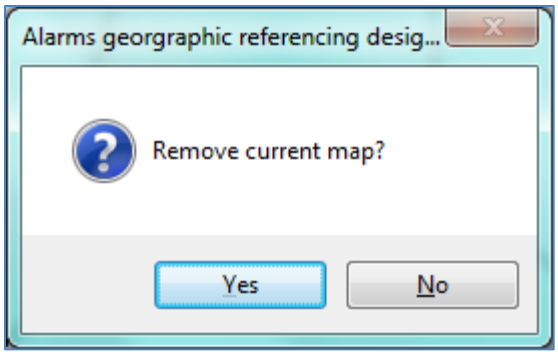


Figure #13.6

Renaming a Map

To rename a map:

1. Select the map and then click **Rename Map.**
2. Enter a new map name in the **Map Name** textbox and click **OK**.

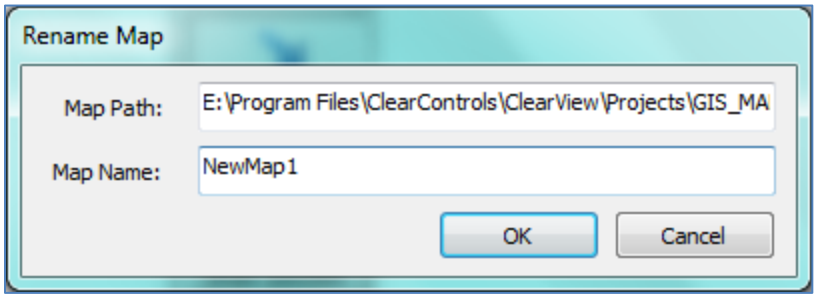


Figure #13.7

Changing Map Path

To change a map path:

3. Select the map and then click **Change Map Path.**
4. Browse the file system to choose new map file and press **Open**.

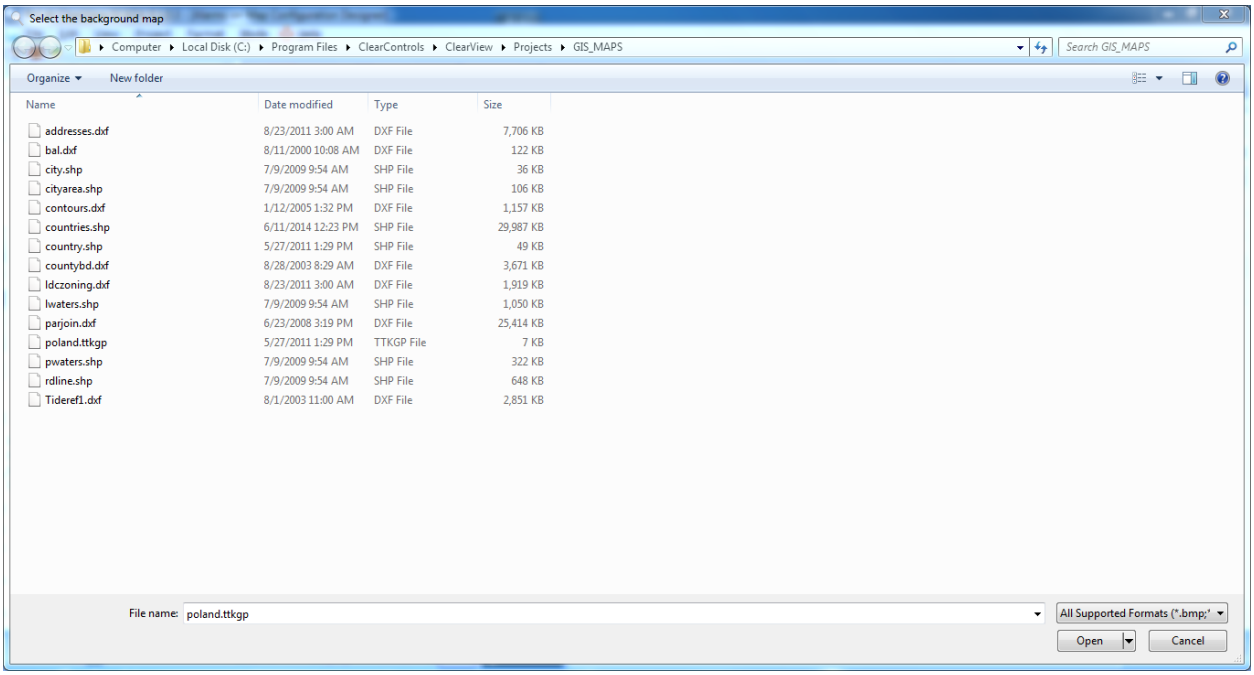


Figure #13.8

Draw Location and Junction

To draw the location, click **Draw Location** on the toolbar

To draw a junction, select the first location, click **Draw Junction,** then click the second location. The line will appear between these two locations.

Figure #13.9

Deleting Location or Junction

Right-click the selected location or junctions, then click **Delete Selected Object.**

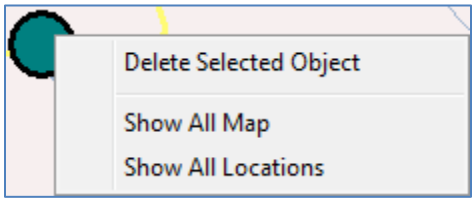


Figure #13.10

Location or Junction Properties

Each Location and Junction has the following properties: Name, Device, Custom and Style.

To change the properties:

1. Select a location or junction.
2. Enter the new **Name**, **Device**, and **Custom** properties, choose required **Style**
3. Click **Save** to save the project.

Note that Name, Device and Custom properties can be any values of **String** type. Those properties can be used in the events generated by double-click on a location or a junction, see **ClearView-SCADA GIS Scripting Functions** below.

Location and Junction styles are configured by changing **Settings**, see below.

Changing Settings

To change the design settings, click **Settings**. User can change settings for Locations, Junctions and Maps.

Locations Tab

User can create multiple styles for Locations and then use them to create new or modify existing locations.

To add new location Style:

1. Switch to **Locations** tab.
2. Press **Add**. A new Style will be added.
3. Specify Style **Name**, **Use as Default**, **Shape**, and other properties.

You can choose one of the standard Shapes like Box, Circle or Triangle or specify a custom picture.

There could be only one Style with **Use as Default** property set to **True**. If you set it to **True** for one style this property will be automatically set to **False** for all other styles.

Note: **Use as Default** property sets the style as a **Global Default Style** that will be used for a map for which the **Map Default Style** is not specified. For example, if you open a new map and add a Location or Junction they will be added with **Global Default Style**. User can also specify a map specific default style, see **Maps Tab** below.

Note: GIS is compatible with standard picture formats like Bmp, Jpeg, Gif, Icon, etc. When using Icons make sure they are a single image Icons; the icon size for **Normal Size** should be 16x16 pixels, icon size for **Selected Size** should be 32x32 pixels. If you choose to use a picture instead of a shape be aware that the size of the location image is defined only by the picture size and there is no border as it is for a shape.

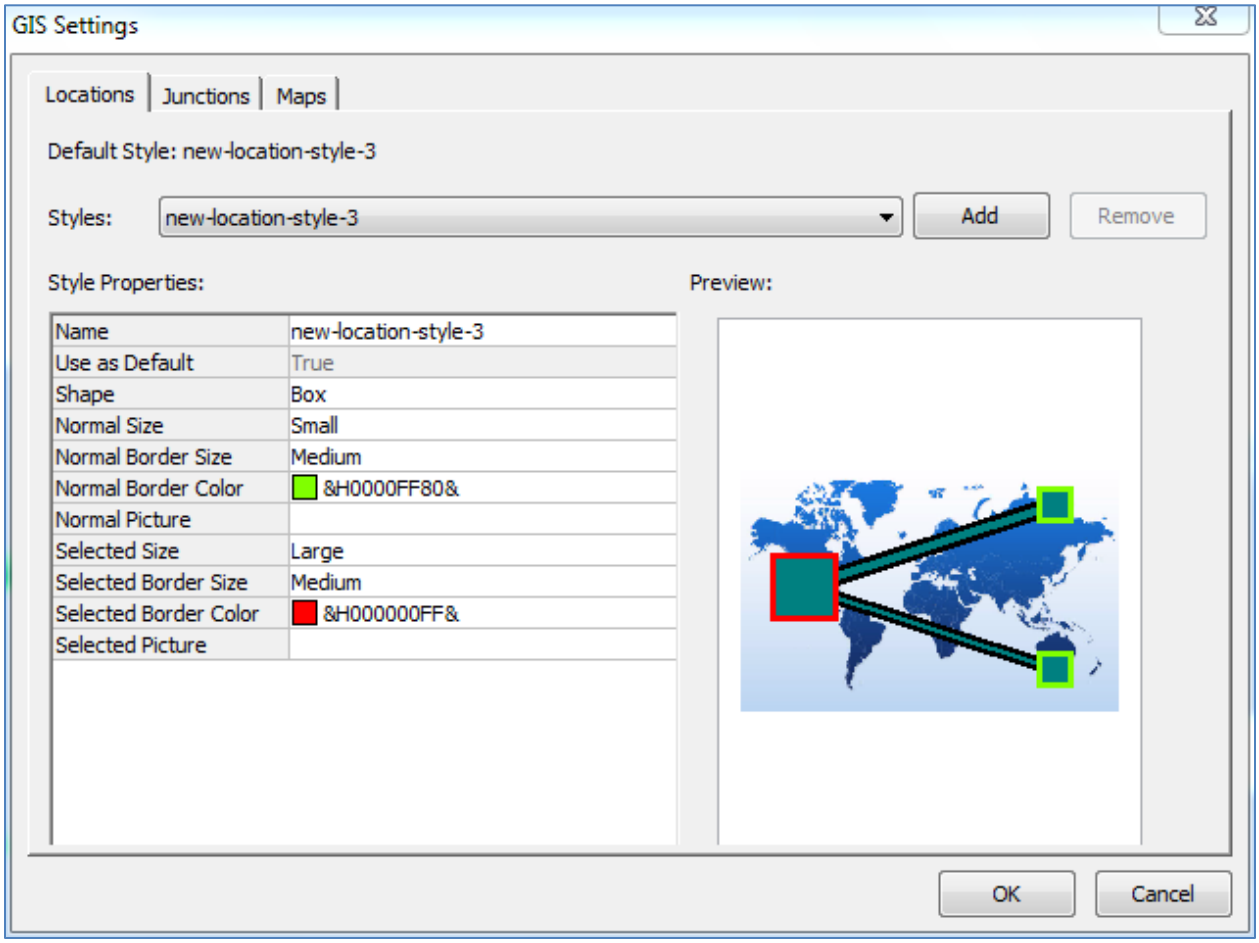


Figure #13.11

Junctions Tab

User can create multiple styles for junctions and then use them to create new or modify existing junctions.

To add new junction Style:

1. Switch to **Junctions** tab
2. Press **Add**. New Style will be added.
3. Specify Style Name, Use as Default, Normal Size, and other properties.

There could be only one Style with **Use as Default** property set to **True**. If you set it to **True** for one style this property will be automatically set to **False** for all other styles.

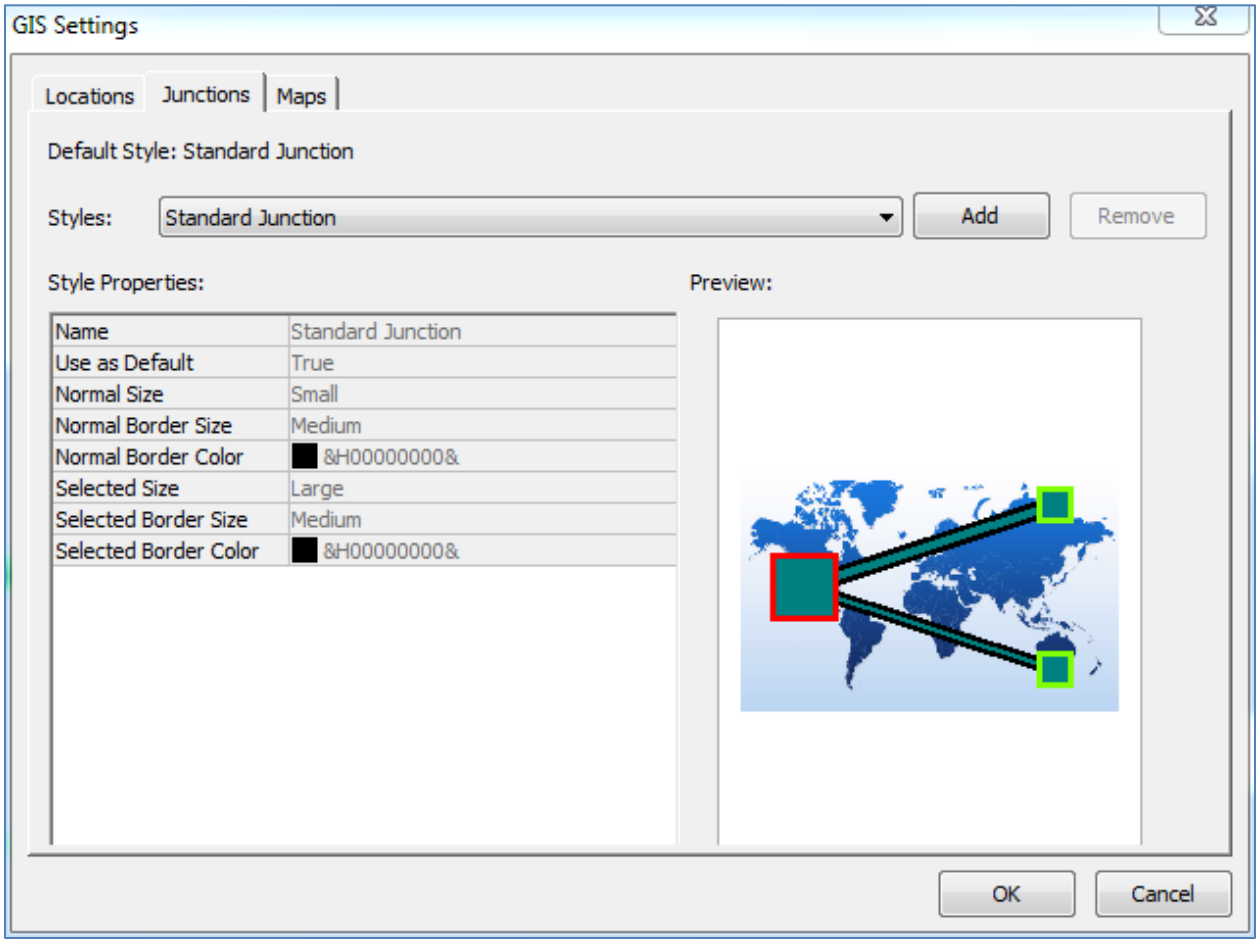


Figure #13.12

Maps Tab

Any time a user creates a new location or junction they will be created with a default style. If the default style is not specified (left as <none>) then the Global Default Style will be used. The Global Default Style can be specified in style's properties on the Location and Junction tabs.

Maps tab allows user to specify and apply the default Location and Junction styles per a map.

To specify a map specific default style:

1. Select a map from the **Maps** drop-down list.
2. Select default location style from **Default Location Style** drop-down list (for selected map only).
3. Select default junction style from **Default Junction Style** drop-down list (for selected map only).
4. If you want the default style to be applied to all locations or junctions on the selected map - press **Apply to All Locations on the Map** or **Apply to All Junctions on the Map.** You will be prompted with a confirmation message box; press **Yes** to confirm the change. Note that actual change will happen after you press **Ok** on the GIS Settings window. The change will not happen of you close the GIS Settings window by pressing **Cancel**.

Warning: Be aware that as the result of using this feature all locations and junctions' styles will be replaced on the selected map.

5. Press Ok to close GIS Setting window.

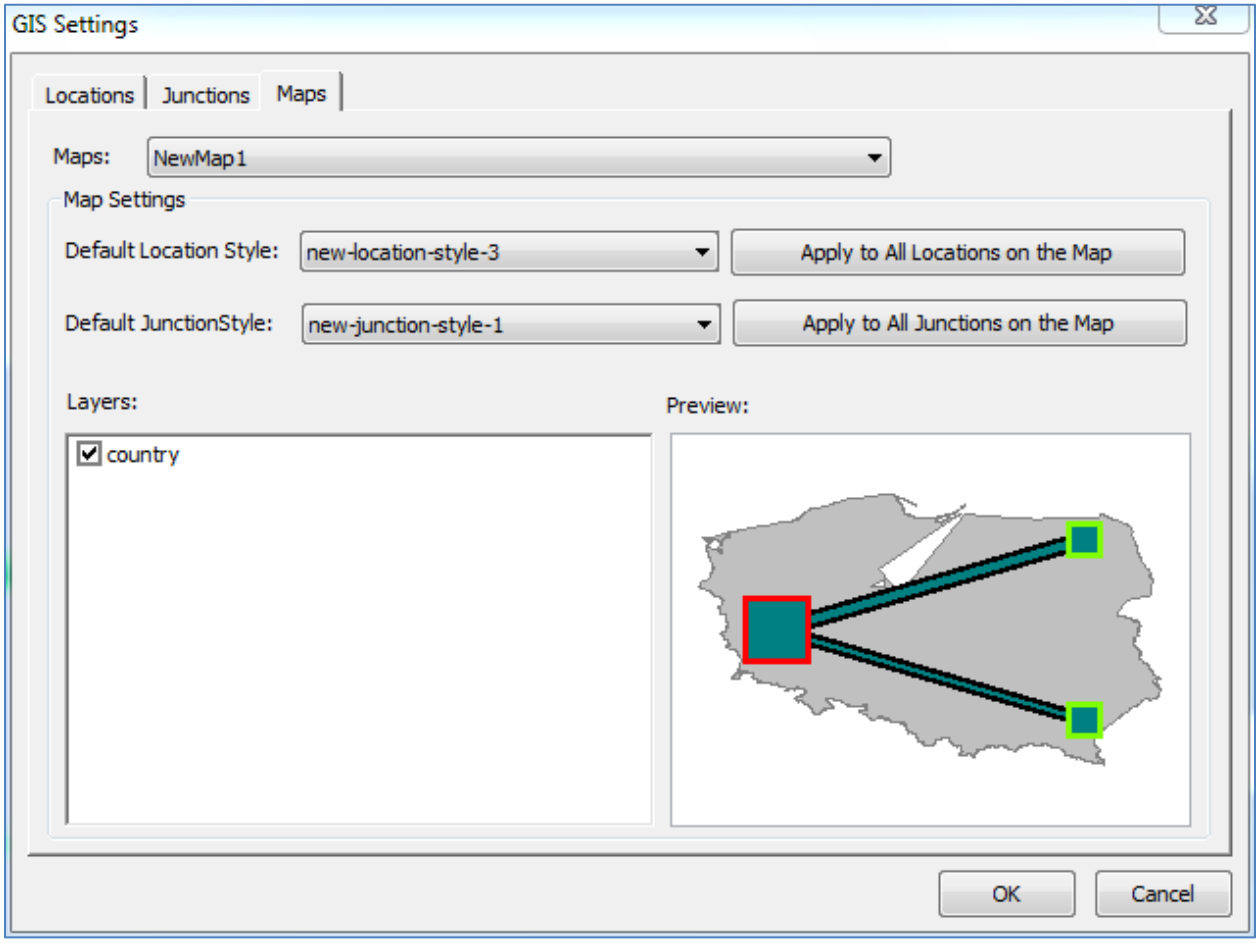


Figure #13.13

Assigning Alarms to Location or Junction

To assign an alarm to a specific location or junction, select the location/junction and click **Assign/Unassign**.

- To add an alarm, select the alarm and click **>**.
- To remove an alarm, select the alarm(s) from the **Selected Alarms** list and click **<**.
- To filter for a specific alarm, enter alarm name in to **Filter** textbox and click **Apply** button.
- To complete the configuration, click **OK.**

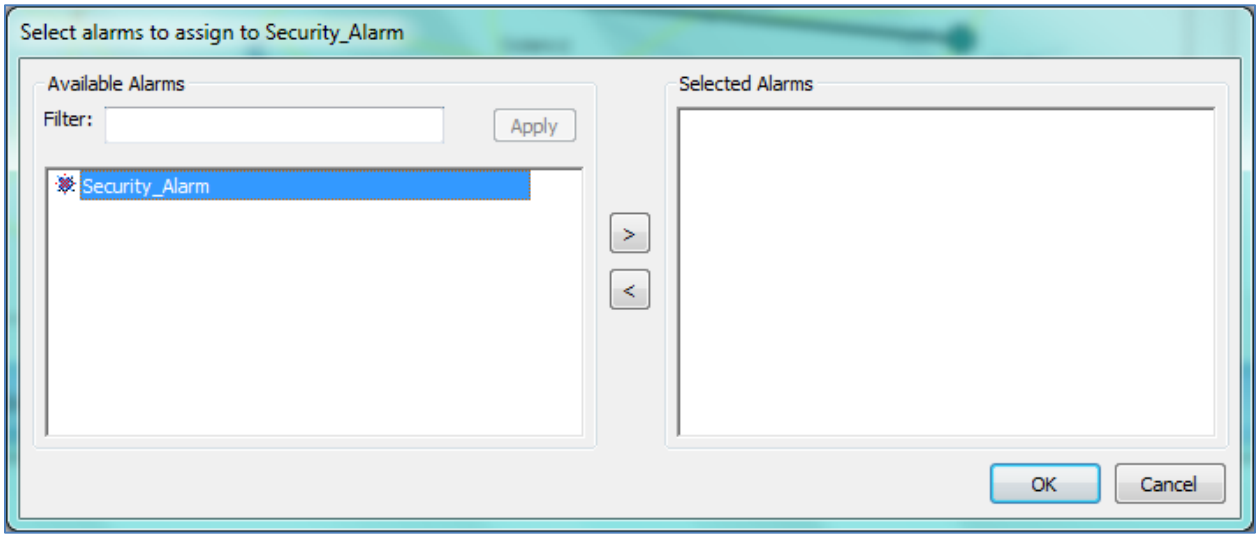


Figure #13.14

Assigning Tags to Alarm Location or Junction

- After the alarm is assigned, click **Configure** to assign tag(s) to a specific location or junction.
- To add tags, select the tag and click **>.**
- To remove tags, select the alarm from **Selected Tags** list and click **<**.
- To filter for a specific tag(s), enter tag name(s) in the Filter text box and click **Apply**.
- To complete configuration, click **OK.**

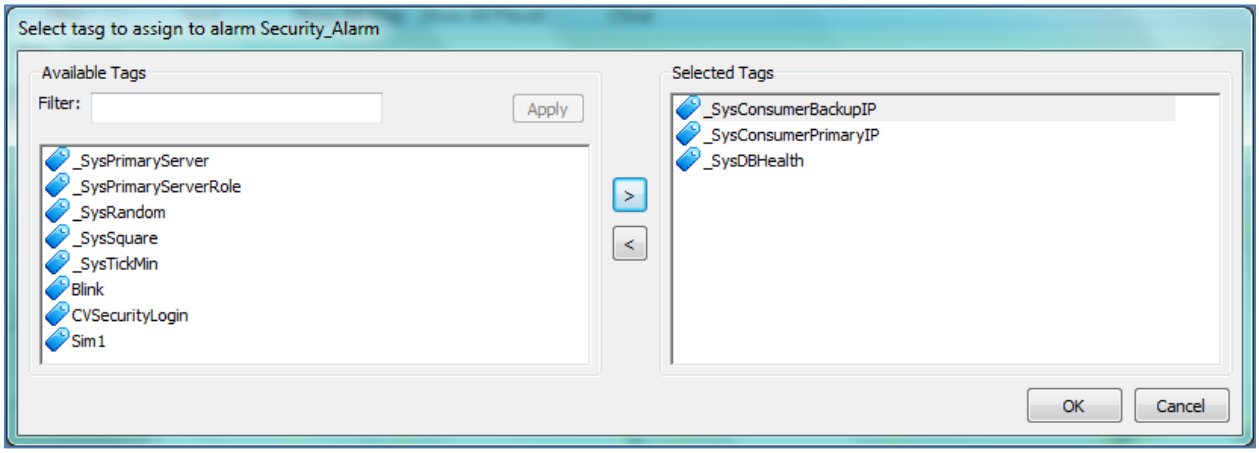


Figure #13.15

GIS Configuration Toolbar

Item	Description
New	Creates a new project. Warning: This will erase existing configuration.
Save	Saves a project.
Show All Map	Displays a full-size map.
Show All Places	Displays map places region.
Close	Closes the GIS configuration interface.

GIS Viewer

Opening the GIS Viewer

To open the GIS Viewer in **Application Explorer,** double-click **GIS Viewer** under **Alarms and Events** folder.

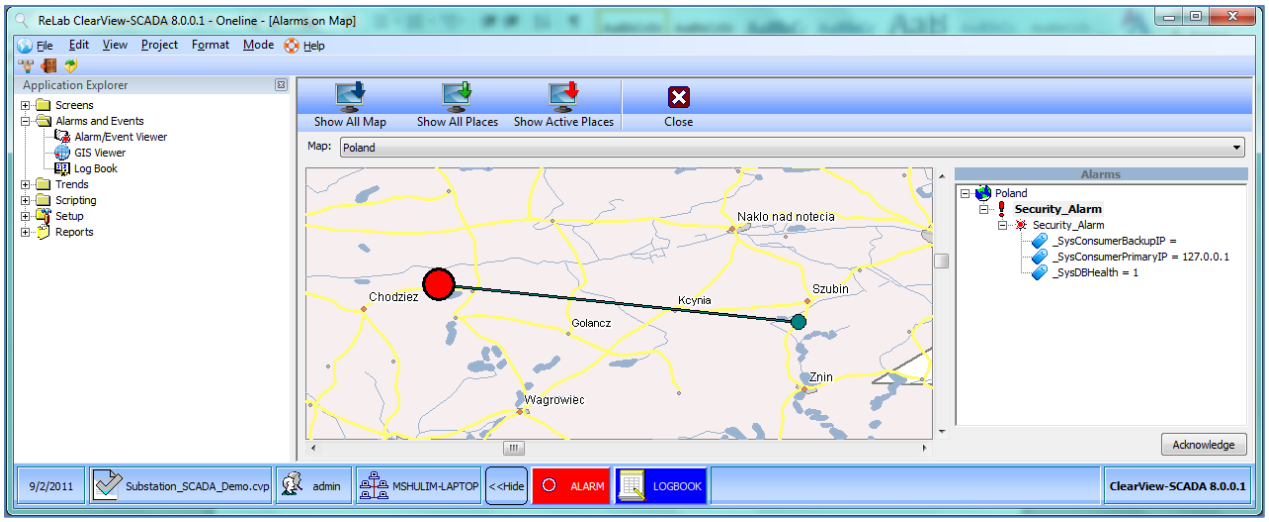


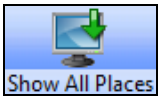
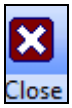
Figure #13.16

Alarm Acknowledgment

Select a specific location or junction and click **Acknowledge**.

If the alarm was already acknowledged the button will display **Reset** instead of **Acknowledge**.

GIS Viewer Toolbar

Item	Description
	Displays full-size map.
	Displays map places region.
	Displays active alarm(s) region.
	Closes GIS Viewer interface.

ClearView-SCADA GIS Scripting Functions

Item	Description	Parameters	Example
Screens.ShowScreen("AlarmsGIS")	Opens GIS configuration interface.	Name of GIS configuration screen	Screens.ShowScreen("AlarmsGIS")
Screens.ShowScreen("AlarmsGISRT")	Opens GIS Viewer.	Name of GIS Viewer Screen	Screens.ShowScreen("AlarmsGISRT")
Project.ShowMap "MyMap"	Opens a specific map.	Map Name	Project.ShowMap "MyMap"
Private Sub Project_OnGISDblClick(ObjectName)	Event is fired when location or junction double-clicked.	Clicked Location or Junction Name	Private Sub Project_OnGISDblClick(ObjectName) 'Place your code here End Sub
Private Sub Project_OnGISDblClickEx(ObjectName, MapName, Device, Custom)	Event is fired when location or junction double-clicked.	Clicked location or junction name, clicked map name, Device and Custom properties of the location or junction.	Private Sub Project_OnGISDblClickEx(ObjectName, MapName, Device, Custom) 'Place your code here

Item	Description	Parameters	Example
			End Sub

Function: GetLocationPosX(MapName, LocName)	Returns location X coordinates	MapName: Name of the map where location is present (String)
		LocName: Name of the location (String)
			If Project.GetLocationPosX ("MapName", "LocationName") > 100.234 Then
			'code here
			End If

Function: GetLocationPosY(MapName, LocName)	Returns location Y coordinates	MapName: Name of the map where location is present (String)
		LocName: Name of the location (String)
			If Project.GetLocationPosY ("MapName", "LocationName") > 100.234 Then
			'code here
			End If

Function: GetCSWKT(MapName)	Returns map coordinate system index (see table below)	MapName: Name of the map (String)	If Project.GetCSWKT ("MapName") = 1 Then
			'code here
			End If

Sub: SetLocationPos(MapName, LocName, X, Y, CSWKT)	Sets location position	MapName: Name of the map (String) LocName: Name of the location (String) X: new "X" coordinates Y: new "Y" coordinates CSWKT: Map coordinate system index (see table below) Note: CSWKT = -1 means not supported coordinate system	Project.SetLocationPos "MapName", "LocationName", 123456.45, 444.555, 234

Fault Location Example:

```
Private Sub Fault_Detection ( )

Dim Map_Name          ' Name of the map
Dim Loc1              ' Name of the first location
Dim Loc2              ' Name of the second location
Dim Loc3              ' Name of the fault location

Dim X1                ' [X] Pole1 position
Dim Y1                ' [Y] Pole1 position
Dim X2                ' [X] Pole2 position
Dim Y2                ' [Y] Pole2 position
Dim X3                ' [X] Fault position
Dim Y3                ' [Y] Fault position
Dim CSWKT             ' Geographical system index
Dim Offset            ' Fault location offset (0-1) in meters or feet

Map_Name = "Map_Name"
```

```
Loc1 = "Pole1"
Loc2 = "Pole2"
Loc3 = "Fault"

CSWKT = Project.GetCSWKT ( Map_Name )
Offset = 0.4657

' Set XY position for Loc1 & Loc2
X1 = Project.GetLocationPosX ( Map_Name, Loc1)
Y1 = Project.GetLocationPosY ( Map_Name, Loc1)
X2 = Project.GetLocationPosX ( Map_Name, Loc2)
Y2 = Project.GetLocationPosY( Map_Name, Loc2)

' Set XY fault position the Fault location
X3 = X1+(X2-X1) * Offset
Y3 = Y1+(Y2-Y1) * Offset

' Set move fault to the position
Project.SetLocationPos "Map_Name", "Fault", X3, Y3, CSWKT
End Sub
```

Map Coordinate System

Coordinate System Description	Index
Not supported	-1
Abidjan_1987	0
Accra	1
Adindan	2
Afgooye	3
Agadez	4
AGD66	5
AGD84	6
Ain_el_Abd	7
Alaskan_Islands	8
Albanian_1987	9
American_Samoa_1962	10
Amersfoort	11
Ammassalik_1958	12
Anguilla_1957	13
Anna_1_1965	14
Antigua_1943	15
Aratu	16
Arc_1950	17
Arc_1960	18
Ascension_Island_1958	19

Coordinate System Description	Index
Astro_1952	20
ATS77	21
Australian_Antarctic	22
Ayabelle_Lighthouse	23
Azores_Central_1948	24
Azores_Central_1995	25
Azores_Occidental_1939	26
Azores_Oriental_1940	27
Azores_Oriental_1995	28
Barbados_1938	29
Batavia	30
Batavia_Jakarta	31
BDA2000	32
Beacon_E_1945	33
Beduaram	34
Beijing_1954	35
Belge_1950	36
Belge_1950_Brussels	37
Belge_1972	38
Bellevue	39
Bermuda_1957	40
Bern_1898_Bern	41
Bern_1938	42
Bissau	43
Bogota_1975	44
Bogota_1975_Bogota	45
Bukit_Rimpah	46
Cadastre_1997	47
Camacupa	48
Camp_Area	49
Camp_Area_Astro	50
Campo_Inchauspe	51
Canton_1966	52
Cape	53
Cape_Canaveral	54
Carthage	55
Carthage_Paris	56
CH1903	57

Coordinate System Description	Index
CH1903Plus	58
Chatham_Island_1971	59
Chatham_Islands_1971	60
Chatham_Islands_1979	61
China_Geodetic_Coordinate_System_2000	62
Chos_Malal_1914	63
CHTRF95	64
Chua	65
Cocos_Islands_1965	66
Combani_1950	67
Conakry_1905	68
Corrego_Alegre	69
Cote_d_Ivoire	70
CSG67	71
Custom_1	72
Custom_2	73
Custom_3	74
Custom_4	75
Custom_5	76
Custom_6	77
Dabola_1981	78
Datum_73	79
Datum_Lisboa_Bessel	80
Datum_Lisboa_Hayford	81
Dealul_Piscului_1930	82
Dealul_Piscului_1970	83
Deception_Island	84
Deir_ez_Zor	85
DGN95	86
DHDN	87
Diego_Garcia_1969	88
Dominica_1945	89
DOS_1968	90
DOS_71_4	91
Douala	92
Douala_1948	93
DRUKREF_03	94
Easter_Island_1967	95

Coordinate System Description	Index
ED50	96
ED50_ED77	97
ED79	98
ED87	99
Egypt_1907	100
Egypt_1930	101
Egypt_Gulf_of_Suez_S_650_TL	102
ELD79	103
EST92	104
EST97	105
Estonia_1937	106
ETRS89	107
European_1979	108
Everest_Bangladesh	109
Everest_India_Nepal	110
Everest_Modified_1969	111
Fahud	112
Fatu_Iva_72	113
FD54	114
FD58	115
Fiji_1956	116
Fiji_1986	117
Fischer_1960	118
Fischer_1968	119
Fischer_Modified	120
fk89	121
Fort_Marigot	122
Fort_Thomas_1955	123
Gan_1970	124
Gandajika	125
Gandajika_1970	126
Garoua	127
GDA94	128
GDBD2009	129
GDM2000	130
GGRS87	131
GR96	132
Graciosa_Base_SW_1948	133

Coordinate System Description	Index
Grand_Cayman_1959	134
Grand_Comoros	135
Greek	136
Greek_Athens	137
Grenada_1953	138
Guadeloupe_1948	139
Guam_1963	140
Gulshan_303	141
GUX_1	142
Guyane_Francaise	143
Hanoi_1972	144
Hartebeesthoek94	145
HD1909	146
HD72	147
Helle_1954	148
Herat_North	149
Hermannskogel	150
Hito_XVIII_1963	151
Hjorsey_1955	152
Hong_Kong_1963	153
Hong_Kong_1963_67	154
Hong_Kong_1980	155
Hough_1960	156
HTRS96	157
Hu_Tzu_Shan_1950	158
ID74	159
IGC_1962_6th_Parallel_South	160
IGCB_1955	161
IGM95	162
IGN_1962_Kerguelen	163
IGN_Astro_1960	164
IGN53_Mare	165
IGN56_Lifou	166
IGN63_Hiva_Oa	167
IGN72_Grand_Terre	168
IGN72_Grande_Terre	169
IGN72_Nuku_Hiva	170
IGRS	171

Coordinate System Description	Index
IKBD_92	172
Indian_1954	173
Indian_1960	174
Indian_1975	175
IRENET95	176
ISN93	177
Israel	178
ISTS_061_1968	179
ISTS_073_1969	180
Iwo_Jima_1945	181
JAD2001	182
JAD69	183
Jamaica_1875	184
JGD2000	185
Johnston_Island_1961	186
Jouik_1961	187
K0_1949	188
Kalianpur_1880	189
Kalianpur_1937	190
Kalianpur_1962	191
Kalianpur_1975	192
Kandawala	193
Karbala_1979	194
Kasai_1953	195
Katanga_1955	196
Kerguelen_Island_1949	197
Kertau_1968	198
Kertau_RSO	199
KKJ	200
KOC	201
Korea_2000	202
Korean_1985	203
Korean_1995	204
Kousseri	205
KUDAMS	206
Kusaie_1951	207
La_Canoa	208
Lake	209

Coordinate System Description	Index
Lao_1993	210
Lao_1997	211
LC5_1961	212
Le_Pouce_1934	213
Leigon	214
LGD2006	215
Liberia_1964	216
Lisbon	217
Lisbon_1890	218
Lisbon_1890_Lisbon	219
Lisbon_Lisbon	220
Little_Cayman_1961	221
LKS92	222
LKS94	223
LKS94_ETRS89	224
Locodjo_1965	225
Loma_Quintana	226
Lome	227
Luxembourg_1930	228
Luzon_1911	229
M_poraloko	230
Madeira_1936	231
Madrid_1870_Madrid	232
Madzansua	233
MAGNA_SIRGAS	234
Mahe_1971	235
Makassar	236
Makassar_Jakarta	237
Malongo_1987	238
Manoca	239
Manoca_1962	240
Marcus_Island_1952	241
Marshall_Islands_1960	242
Martinique_1938	243
Massawa	244
Maupiti_83	245
Mauritania_1999	246
Merchich	247

Coordinate System Description	Index
Mexican_Datum_of_1993	248
MGI	249
MGI_1901	250
MGI_Ferro	251
Mhast	252
Mhast_1951	253
Mhast_offshore	254
Mhast_onshore	255
Midway_1961	256
Minna	257
MOLDREF99	258
Monte_Mario	259
Monte_Mario_Rome	260
Montserrat_1958	261
Moorea_87	262
MOP78	263
Mount_Dillon	264
Moznet	265
NAD27	266
NAD27_76	267
NAD27_CGQ77	268
NAD27_Michigan	269
NAD83	270
NAD83_CSRS	271
NAD83_CSRS98	272
NAD83_HARN	273
NAD83_NSRS2007	274
Nahrwan_1934	275
Nahrwan_1967	276
Nakhl_e_Ghanem	277
Naparima_1955	278
Naparima_1972	279
NDG_Paris	280
NEA74_Noumea	281
New_Beijing	282
NGN	283
NGO_1948	284
NGO_1948_Oslo	285

Coordinate System Description	Index
Nord_Sahara_1959	286
Nord_Sahara_1959_Paris	287
Nouakchott_1965	288
NSWC_9Z_2	289
NTF	290
NTF_Paris	291
NZGD2000	292
NZGD49	293
Observ_Meteorologico_1939	294
Observatario	295
Old_Hawaiian	296
Oman	297
OS_SN_80	298
OSGB_1936	299
OSGB70	300
OSNI_1952	301
Padang	302
Padang_Jakarta	303
Palestine_1923	304
Pampa_del_Castillo	305
PD_83	306
Perroud_1950	307
Petrels_1972	308
Phoenix_Islands_1966	309
Pico_de_Las_Nieves	310
Pico_de_las_Nieves_1984	311
Pitcairn_1967	312
Pitcairn_2006	313
Point_58	314
Pointe_Noire	315
Popular_Visualisation_CRS	316
Porto_Santo	317
Porto_Santo_1995	318
POSGAR	319
POSGAR_94	320
POSGAR_98	321
Principe	322
PRS92	323

Coordinate System Description	Index
PSAD56	324
PSD93	325
PTRA08	326
Puerto_Rico	327
Pulkovo_1942	328
Pulkovo_1942_58	329
Pulkovo_1942_83	330
Pulkovo_1995	331
PZ_90	332
Qatar_1948	333
Qatar_1974	334
QND95	335
Qornoq	336
Qornoq_1927	337
Rassadiran	338
RD_83	339
REGCAN95	340
REGVEN	341
Reunion	342
Reunion_1947	343
Reykjavik_1900	344
RGF93	345
RGFG95	346
RGM04	347
RGNC_1991	348
RGNC91_93	349
RGPF	350
RGR92	351
RGRDC_2005	352
RGSPM06	353
RRAF_1991	354
RSRGD2000	355
RT38	356
RT38_Stockholm	357
RT90	358
S_JTSK	359
S_JTSK_05	360
S_JTSK_05_Ferro	361

Coordinate System Description	Index
S_JTSK_Ferro	362
S42_Hungary	363
SAD69	364
Saint_Pierre_et_Miquelon_1950	365
Samboja	366
Santo_1965	367
Santo_DOS_1965	368
Sao_Braz	369
Sao_Tome	370
Sapper_Hill_1943	371
Schwarzeck	372
Scoresbysund_1952	373
Segara	374
Segara_Jakarta	375
Segora	376
Selvagem_Grande	377
Selvagem_Grande_1938	378
Serindung	379
Sierra_Leone_1924	380
Sierra_Leone_1960	381
Sierra_Leone_1968	382
SIRGAS_1995	383
SIRGAS_2000	384
SLD99	385
Slovenia_1996	386
Solomon_1968	387
South_Asia_Singapore	388
South_Georgia_1968	389
South_Yemen	390
SREF98	391
St_George_Island	392
St_Helena_1971	393
St_Kitts_1955	394
St_Lawrence_Island	395
St_Lucia_1955	396
St_Paul_Island	397
St_Vincent_1945	398
ST71_Belep	399

Coordinate System Description	Index
ST84_Ile_des_Pins	400
ST87_Ouvea	401
Sudan	402
SVY21	403
SWEREF99	404
Tahaa_54	405
Tahiti_52	406
Tahiti_79	407
Tananarive	408
Tananarive_Paris	409
TC_1948	410
Tern_Island_1961	411
Tete	412
Timbalai_1948	413
TM65	414
TM75	415
Tokyo	416
Tokyo_1892	417
Trinidad_1903	418
Tristan_1968	419
TUREF	420
TWD67	421
TWD97	422
Unknown_datum_based_upon_the_Airy_1830_ellipsoid	423
Unknown_datum_based_upon_the_Airy_Modified_1849_ellipsoid	424
Unknown_datum_based_upon_the_Australian_National_Spheroid	425
Unknown_datum_based_upon_the_Authalic_Sphere	426
Unknown_datum_based_upon_the_Average_Terrestrial_System_1977_ellipsoid	427
Unknown_datum_based_upon_the_Bessel_1841_ellipsoid	428
Unknown_datum_based_upon_the_Bessel_Modified_ellipsoid	429
Unknown_datum_based_upon_the_Bessel_Namibia_ellipsoid	430
Unknown_datum_based_upon_the_Clarke_1858_ellipsoid	431
Unknown_datum_based_upon_the_Clarke_1866_ellipsoid	432
Unknown_datum_based_upon_the_Clarke_1866_Michigan_ellipsoid	433
Unknown_datum_based_upon_the_Clarke_1880_Arc_ellipsoid	434
Unknown_datum_based_upon_the_Clarke_1880_Benoit_ellipsoid	435
Unknown_datum_based_upon_the_Clarke_1880_ellipsoid	436
Unknown_datum_based_upon_the_Clarke_1880_IGN_ellipsoid	437

Coordinate System Description	Index
Unknown_datum_based_upon_the_Clarke_1880_RGS_ellipsoid	438
Unknown_datum_based_upon_the_Clarke_1880_SGA_1922_ellipsoid	439
Unknown_datum_based_upon_the_Everest_1830_1937_Adjustment_ellipsoid	440
Unknown_datum_based_upon_the_Everest_1830_1962_Definition_ellipsoid	441
Unknown_datum_based_upon_the_Everest_1830_1967_Definition_ellipsoid	442
Unknown_datum_based_upon_the_Everest_1830_1975_Definition_ellipsoid	443
Unknown_datum_based_upon_the_Everest_1830_Definition_ellipsoid	444
Unknown_datum_based_upon_the_Everest_1830_Modified_ellipsoid	445
Unknown_datum_based_upon_the_GEM_10C_ellipsoid	446
Unknown_datum_based_upon_the_GRS_1967_ellipsoid	447
Unknown_datum_based_upon_the_GRS_1980_ellipsoid	448
Unknown_datum_based_upon_the_Helmert_1906_ellipsoid	449
Unknown_datum_based_upon_the_Indonesian_National_Spheroid	450
Unknown_datum_based_upon_the_International_1924_ellipsoid	451
Unknown_datum_based_upon_the_Krassowsky_1940_ellipsoid	452
Unknown_datum_based_upon_the_NWL_9D_ellipsoid	453
Unknown_datum_based_upon_the_OSU86F_ellipsoid	454
Unknown_datum_based_upon_the_OSU91A_ellipsoid	455
Unknown_datum_based_upon_the_Plessis_1817_ellipsoid	456
Unknown_datum_based_upon_the_Struve_1860_ellipsoid	457
Unknown_datum_based_upon_the_War_Office_ellipsoid	458
Unknown_datum_based_upon_the_WGS_72_ellipsoid	459
Unknown_datum_based_upon_the_WGS_84_ellipsoid	460
Unspecified_datum_based_upon_the_Clarke_1866_Authalic_Sphere	461
Unspecified_datum_based_upon_the_GRS_1980_Authalic_Sphere	462
Unspecified_datum_based_upon_the_Hughes_1980_ellipsoid	463
Unspecified_datum_based_upon_the_International_1924_Authalic_Sphere	464
Vanua_Levu_1915	465
Vientiane_1982	466
Viti_Levu_1912	467
Viti_Levu_1916	468
VN_2000	469
Voirol_1875	470
Voirol_1875_Paris	471
Voirol_1879	472
Voirol_1879_Paris	473
Wake_Eniwetok_1960	474
Wake_Island_1952	475

Coordinate System Description	Index
Walbeck	476
WGS_66	477
WGS_72	478
WGS_72BE	479
WGS_84	480
Xian_1980	481
Yacare	482
Yemen_NGN96	483
Yoff	484
Zanderij	485

GIS Zoom and Pan

Item	Description
Zoom-in	To zoom in, hold Shift key and draw a rectangle from left to right using mouse.
Zoom-out	To zoom out, hold Shift key and draw a rectangle from right to left using mouse
Pan	To pan, hold the left mouse button and move the mouse to the Pan position.

SECTION 14

Graphical Reports

ClearView Graphical Reports (CVGR) feature is designed for developing chart-like reports and adding those reports into ClearView client screens. The ClearView Graphical Reports Feature is an option for ClearView Client; it is not a standard feature.

CVGR consists of three executables:

- **Report Editor (CVGraphReports.exe)** – Report Editor is a design tool providing GUI for graphical report development;
- **Report Control (PrjGraphRepCtrl.ocx)** – Report Control is an ActiveX control that may be placed into any ActiveX container (ClearView Client screen, in particular) and makes possible displaying and printing graphical reports;
- **Runtime Report Server (GraphRepServer.exe)** – Runtime Report Server is out-of-process ActiveX server used by PrjGraphRepCtrl.ocx for asynchronous (non-blocking) execution of long lasting operations.

Graphical Reports are stored in "report books." These are XML files of specific structure containing definitions of one or several reports. Reports within report book are identified by name and assigned to report book folders. The report book folders themselves are organized into tree-like structure similar to Windows file system. A report book file's default extension is *.grb.xml.

Registering ClearView Graphical Reports

1. In the Graphical Reports dialog box, click **Register**.

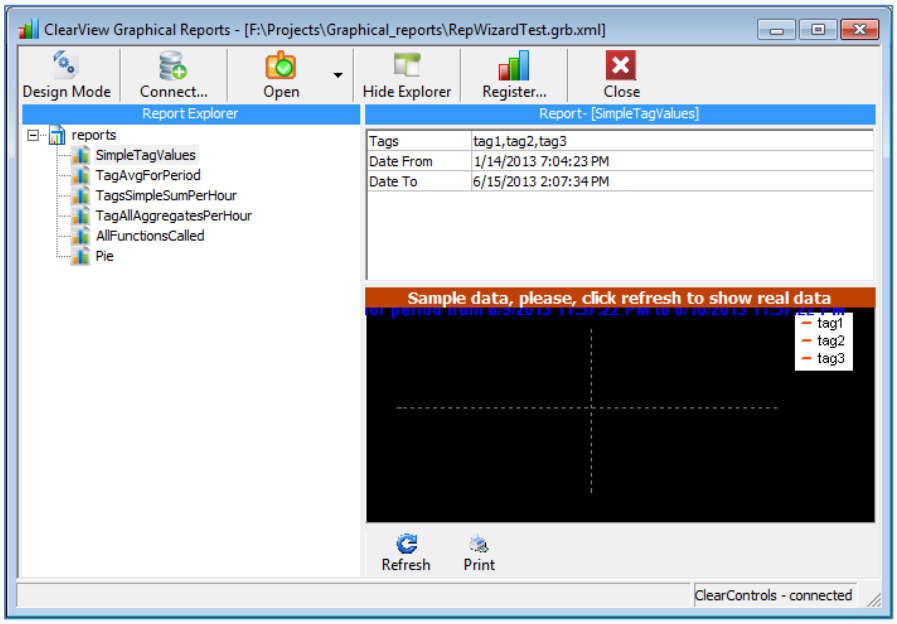


Figure #14.1

2. Copy the **Customer ID** and provide this code to software provider.

Figure #14.2

3. Enter the key code (license) in the **Key Code** field and click **Register**.

Report Editor

When Report Editor starts its user interface, as shown in Figure #14.1. Here you can open a report book, select required report, set required values for report parameters, as well as build and print the report.

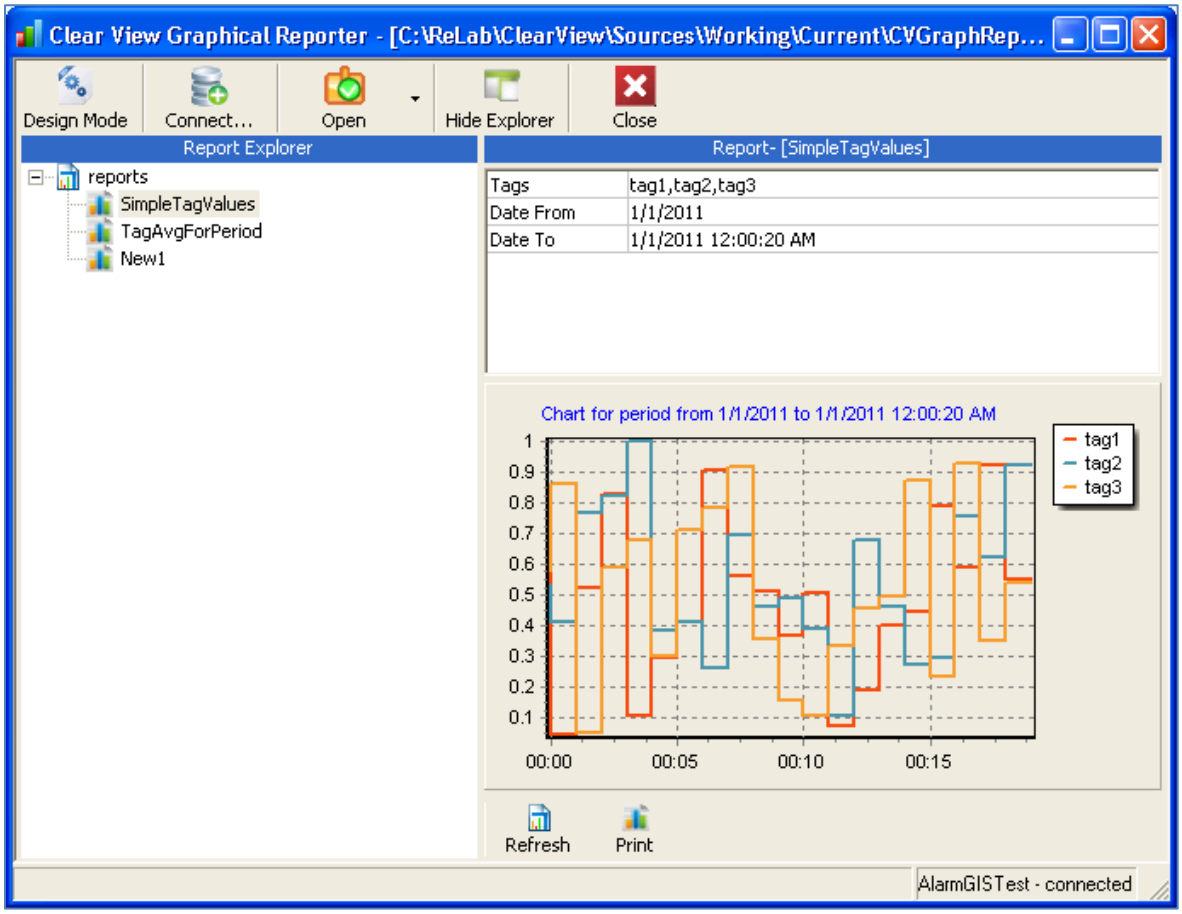


Figure #14.3

- **Open report book**. Click **Open** and select a report book file. The drop-down list shows recently opened report books. When Report Editor starts, the most recently opened report book is opened automatically. In case that report book does not exist, an empty report book appears.
- **Select required report**. Using the Report Explorer panel navigate to the required report. The Report Explorer panel may be shown/hidden using **Hide Explorer**.
- **Set required values of parameters**. Use parameters control over the chart in the right panel for this purpose.
- **Provide connection string**. Commonly, reports use information from the database and require database connection information (connection string). Usually that would be an ODBC data source name (DSN). Connection string is set/changed using "Connect" button. The database connection is not kept open, connectivity is checked when connection string is set/changed and corresponding status is indicated on the status bar of the Report Editor. When editor is started the most recent connection string is used automatically.
- **Build the report**. Click **Refresh** to build the report using the selected parameters.
- **Print the report**. Click **Print** to print current chart. This will show the Print Preview dialog box where the user may select print settings and send the report to a printer.

In order to create new reports or update existing reports the user should switch to Design mode by pressing **Design Mode**.

Designing Reports Concept

Report design consists of two major parts:

Visual Appearance

The chart types are:

- Area
- Bar
- Line
- Pie

Selecting a chart type may provide additional settings such as colors, fonts, sizes, scale, etc.

Report Data

Below is an algorithm for supplying the chart with data. That algorithm is provided in the form of a VBScript with a specific structure. When the user requests to build the report, the script is executed using report parameters provided by the user. The script should have code that fills the chart with the necessary data. The details will be provided in section **3.0 Understanding Script**.

Report design is stored in the report book under tag <item> with nested tags <script> and <chart> like shown below:

```
<report-book name="reports">
...
<item name="SimpleTagValues" type="report">
                    <script> ......... </script>
                    <chart> ............ </chart>
</item>
...
</report-book>
```

The chart setting and script text are stored in binary form and are not available for manual editing. Report Editor must be used to design, build and edit reports

General Functions

In the Design mode Report Editor has extra functionality.

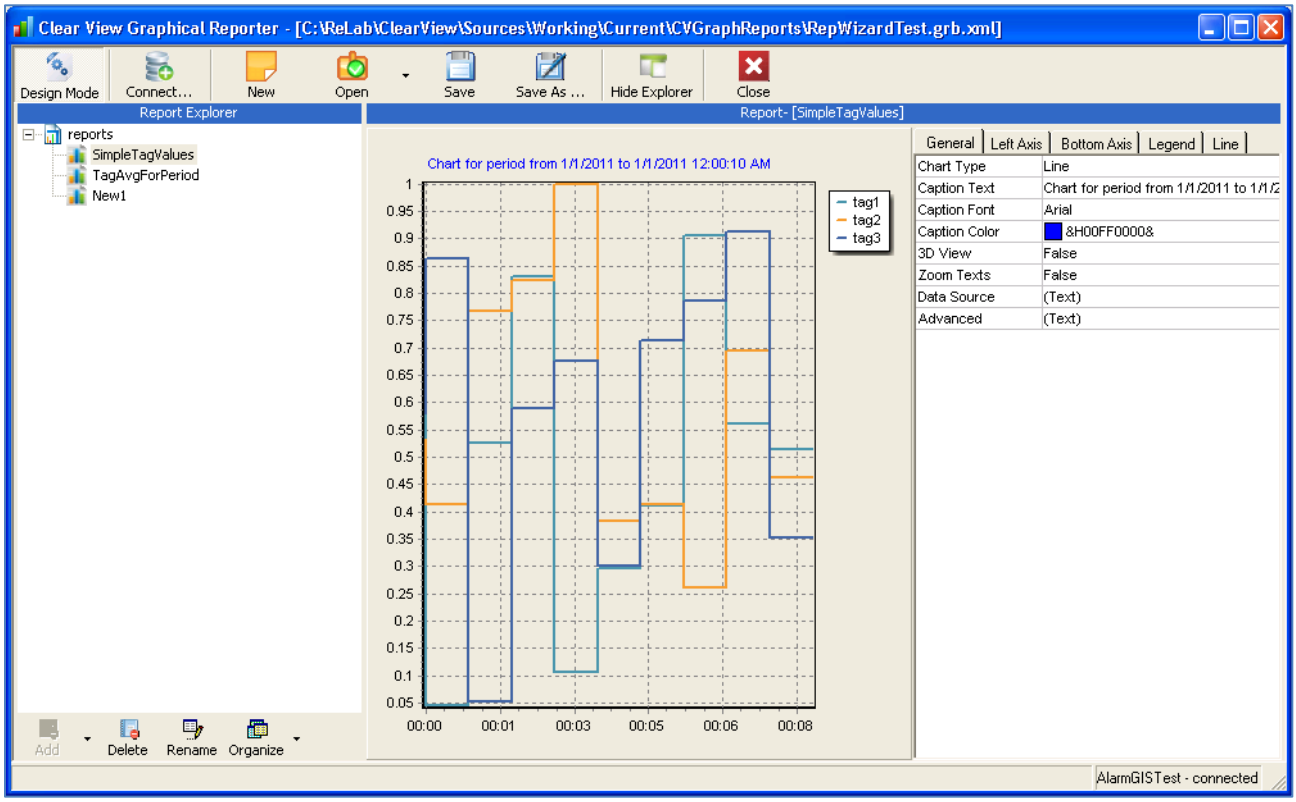


Figure #14.4

Main toolbar has additional buttons for creating new and saving current report book.

The Report Explorer panel has toolbar with buttons for manipulating report book contents (adding/deleting/renaming/moving reports and folders)

- **Add** adds report to the current folder. The dropdown list provides possibility to choose between adding report or subfolder.
- **Delete** deletes selected item from the report book.
- **Rename** renames selected item. A check is performed for providing name correctness: name must be unique (case insensitive), start with letter, and contain letters, digits and underscore only.
- **Organize** is intended for moving reports between folders using the Cut/Copy/Paste approach. It has not yet been implemented.
- An additional panel is shown that is intended for defining currently selected report.

Defining Chart Settings

Chart settings are changed with the Properties panel, located to the right of the chart in the Design mode. The panel shows actual appearance of the chart. Initially the chart data are randomly generated to provide a visual example of the chart appearance. This will change as the data source for the chart is added.

Use of the properties panel is straightforward the property names are self-explanatory, all changes are immediately applied to the chart.

Last tab shows properties specific for that specific chart type.

- Property **Advanced** on the tab **General** tab is used for showing universal Chart Edit dialog box. This dialog box provides rich list of settings that makes it possible for very fine-tuning of the chart.
- Property **Data Source** is used for defining VB Script. The details are provided below.

Understanding Script

Creating Simple Reports

In a simple case the report requires just single function BuildReport(params) to be implemented. This function is called by the Report framework from 3 places. The value passed in the *params* argument depends upon the context where BuildReport is called.

Script has access to the built-in object ChartController. This object has a set of methods facilitating report creation. The main method is FillSeries. It has 3 arguments:

- **SeriesIndex**. When BuildReport is called by the framework, the chart controlled by ChartController has single series with index 0. At this point, FillSeries may be called with SeriesIndex = 0 (in this case an existing series will be filled with supplied data) or with SeriesIndex = 1 (in this case a new series will be created first by cloning series 0, and then series 1 will be filled with data). When there are 2 series in the chart the method, FillSeries may be called with SeriesIndex = 2. Here, again, a new series will be first created from series 0 and then filled with data, etc. In general, SeriesIndex should point at available series of series next to the last available series.
- **SeriesColor**. Color used to draw the series. If -1 is passed for this argument, then color will be selected automatically by some default rule.
- **SeriesData**. SeriesData is a two-dimensional array of series points. In case of Pie, the point is (<numeric value>, <textual label>), for the rest of the report types the point is (<numeric X-value>, <numeric Y-value>).

Example 1 – Hard-Coded Pie Report

The script below builds pie with slices representing Fibonacci numbers:

```
Function BuildReport(params)
  ReDim arrChartData(6,2)
  arrChartData(0,0) = 1
  arrChartData(0,1) = "slice0"
  arrChartData(1,0) = 1
  arrChartData(1,1) = "slice1"
  For i = 2 To UBound(arrChartData)
        arrChartData(i,0) = arrChartData(i - 2,0) + arrChartData(i - 1,0)
        arrChartData(i,1) = "slice" & i
  Next
  ChartController.FillSeries 0, -1, arrChartData

End Function
```

Result of the script execution for hard-coded Pie Report.

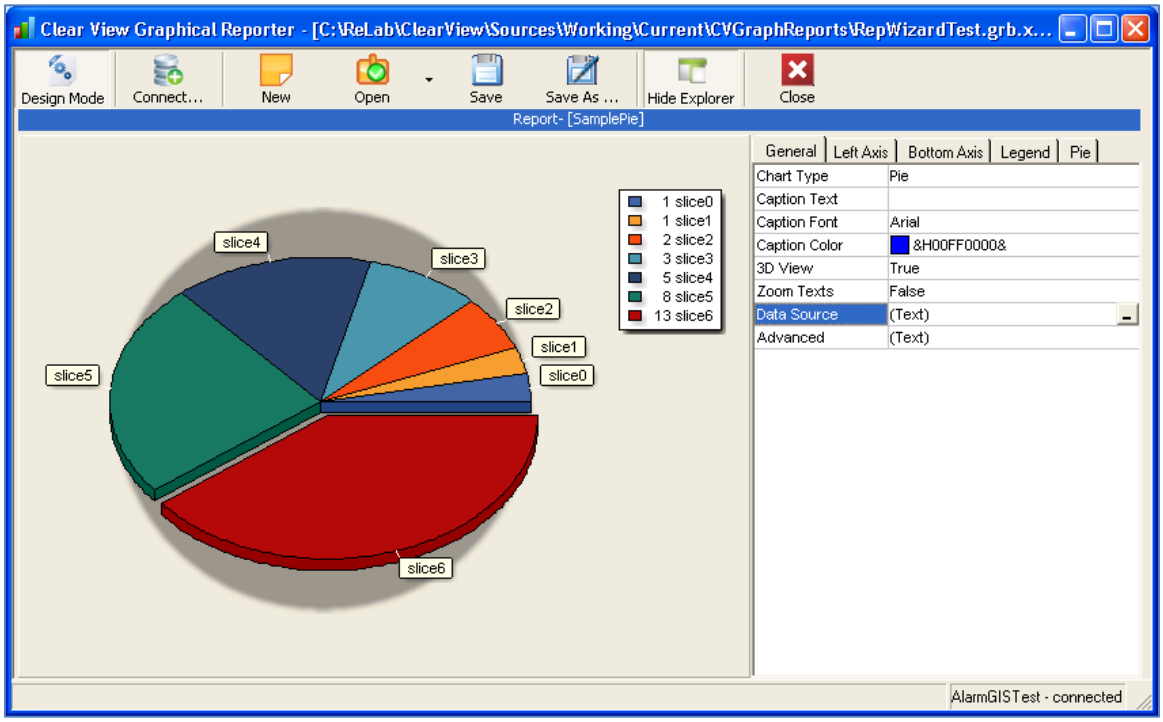


Figure #14.5 – Hard-Coded Pie Report

Example 2 – Hard-Coded Line Report

The script below builds 3 straight lines under different angles to X axis.

```
Function BuildReport (params)
ReDim arrChartData (9, 2)

For i = 0 To 9
arrChartData(i,0) = i
Next

For i = 0 To 9
arrChartData (i,1) = i
Next
ChartController.FillSeries 0, -1, arrChartData

For i = 0 To 9
arrChartData(i,1) = 2 * i
Next
ChartController.FillSeries 1, -1, arrChartData

For i = 0 To 9
arrChartData(i,1) = 3 * i
Next
ChartController.FillSeries 2, -1, arrChartData
```

End Function

The results of the script execution for hard-coded Line report are shown in Figure #14.6

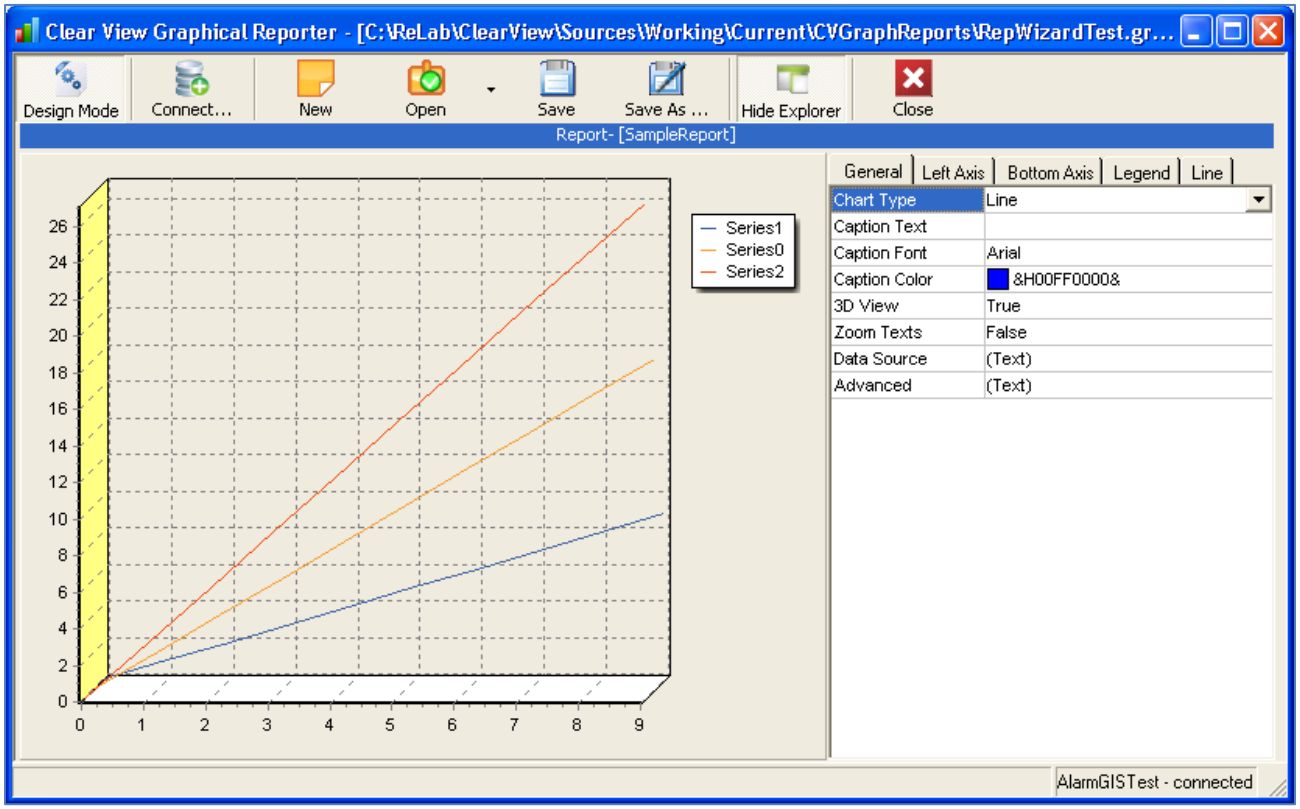


Figure #14.6 – Hard-Coded Line Report

The same result may be represented in different ways, by changing the chart settings. For the Hard-Coded Line Report, the stairs function under the Line tab is changed to true, generating the chart in Figure #14.7.

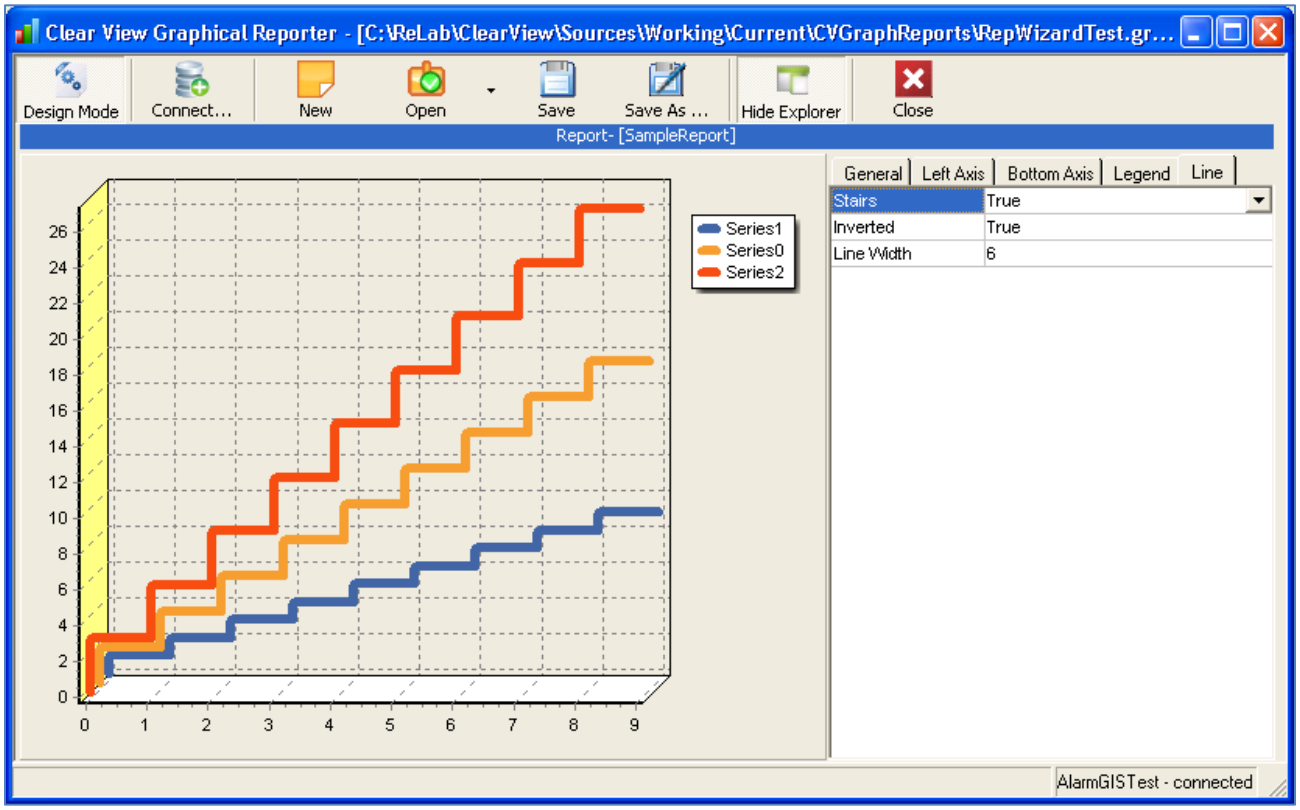


Figure #14.7 – Hard-Coded Line Report with Stairs Chosen as True

Understanding Report Parameters

The Simple Report scripts have limited usefulness because they always produce the same result. In order to provide flexibility the script should use values passed in the *parameters* argument.

The use of parameters is explained below. Still, it is worth mentioning that, even if parameters are not used, there may be approaches for creating useful reports. For instance, the script may read data from files created a day before the current time. An external system may create such files, while the report would show data for the previous day.

As it was mentioned before, the script function BuildReport may be called from 3 places.

- In the method Refresh of the ActiveX control GraphRepCtrl from PrjGraphRepCtrl.ocx the report parameter values are passed in variable list arguments. It is responsibility of the user to pass the values that the report expects.
- When **Refresh** is clicked in the Report Editor. In this case the values are taken from the parameters panel and passed as an array of values. The order of values is like in the GUI.

- When Refresh Chart is clicked on the Script Editor form.

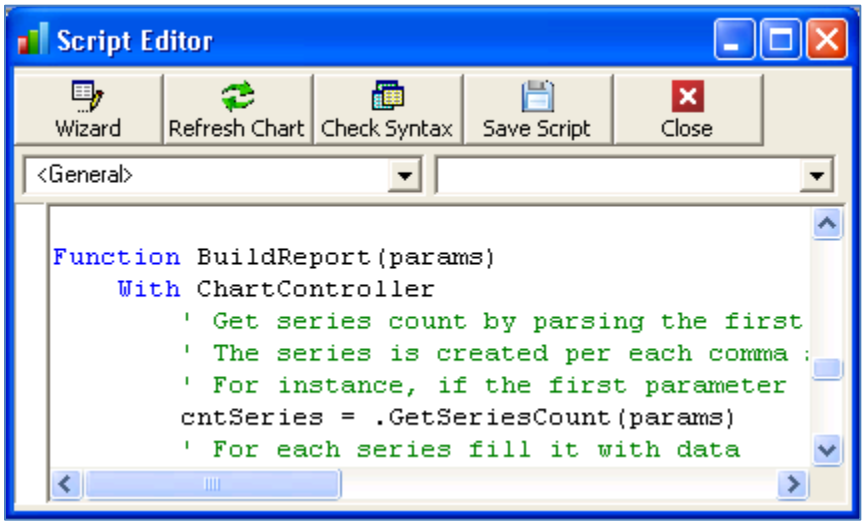


Figure #14.8 – Script Editor Dialog Box

In this case the default parameter values are passed.

For **Refresh** and **Refresh Chart** function to execute, the script must implement procedure GetParamDefs (Names, Captions, Type, DefVals, and DefValTexts). The procedure must assign to the arguments the arrays as described below.

- **Names (i)** – Name of the parameter used to identify it in different places, should be unique and case insensitive way.
- **Captions (i)** – Text shown in the parameters panel of the Report Editor
- **Type(i)** – Parameter type; it may be 0 – number, 1 – string, 2 – datetime
- **DefVals(i)** – Default value of the parameter; it may be any valid VB Script expressions
- **DefValTexts(i**) – It is essentially DefVals(i) but taken into quotes, it is used by GUI to show expression formula in contrast to DefVals(i) that are assigned the result of expression evaluation.

Example 3 – Use of GetParamDefs Function

Example GetParamDefs code is shown below. It defines 3 parameters: list of tags, start date, and end date. End date equals current date, while start date equals 1 day before the current date.

```
Sub GetParamDefs (Names, Captions, Types, DefVals, DefValTexts)
Names = Array ("TagName", "DateFrom", "DateTo")
Captions = Array ("Tags", "Date From", "Date To")
Types = Array (1, 2, 2)
DefVals = Array ("tag1, tag2, tag3", DateDiff (d, -1, Now), Now)
DefValTexts = Array ("tag1, tag2, tag3", "DateDiff (d, -1, Now)", "Now")
End Sub
```

Accessing Database Data

Script may access data in a database immediately using any automation objects providing database connectivity, ADODB in particular. However Report Framework provides functionality simplifying that task for the purpose of the reports development.

Using Report Framework to access data in a database

- Create SQL Select query for getting required data.
- Add parameters to the query, if necessary, in form @<parameter name>~. The same parameter may be used in several places of the query
- In the BuildReport implementations call the method ChartController.RetreiveData. That method has the following arguments.
    - SQL query text
    - SQL query parameter values
    - Output array of the query execution result
    - Optional parameter defining the way the parameter values are passed in the first argument should be mapped to the query parameters. If True then the values corresponds to the parameters as provided by GetParamDefs function, if false – then the values correspond to the parameters from SQL query. Default value is True.

SQL query parameter must have names from the list provided by GetParamDefs function.

Connection string is used by the RetreiveData function and defined in the Report Editor.

Note: There is one special case of SQL query parameter. If parameter name starts with dollar sign ($) the value of this parameter is considered to be comma separated list of values, and SQL query is expanded before being executed. This is useful in case of SQL clause IN. The construct IN(@$MyParam~) is expanded the following way – if $MyParam value is "val1,val2…valN" the result is IN("val1","val2",…,"valN")

Example 4 – Using RetreiveData Function

Below is SQL query example that returns sequence of values of the specified tag for specified time range.

```
SQL = "SELECT DateTimeStamp, HistoryValue " & vbNewLine _
& "FROM History " & vbNewLine _
& "WHERE Tag = @TagName~ " & vbNewLine _
& " AND DateTimeStamp >= @DateFrom~ AND DateTimeStamp <= @DateTo~ " & vbNewLine _
& "ORDER BY DateTimeStamp"
```

If this query is used, then there are two options for using RetreiveData function:

1. Create an array, fill it with values, and pass that array to RetreiveData function. The last parameter should be equal False in this case. The function GetParamDefs is not necessary to have in this case. The order of the parameter values must correspond to the parameters appearance in the query.

```
Sub BuildReport(param)
    …
    ReDim NewParams (2)
    NewParams (0) = "Tag1"
    NewParams (1) = Now
    NewParams (2) = DateDiff (d, -1, Now)
    ChartController.RetreiveData SQL, NewParams, arrResult, False
    …
End Sub
```

2. Write function GetParamDefs in the script that includes among other parameters all query parameters. Then, when calling GetParamDefs, pass it *params* for the first argument and True for the last argument (or just skip the last argument).

```
Sub GetParamDefs (Names, Captions, Types, DefVals, DefValTexts)
        Names = Array ("DateTo", "SomeParam1", "DateFrom", "TagName")
        Captions = Array ("Date To", "Some Param 1", "Date From", "Tags")
        Types = Array (2, 1, 2, 1)
        DefVals = Array (Now, "Def Val of Some Param", DateDiff (d, -1, Now), "Tag1")
        DefValTexts = Array ("Now", "Def Val of Some Param", "DateDiff (d, -1, Now)", "Tag1")
End Sub

Sub BuildReport(param)
        …
        ChartController.RetreiveData SQL, params, arrResult
        …
End Sub
```

Data Aggregation

In the special case where the data represent time sequence of numeric values that data may be processed for obtaining a sequence of the aggregated values; The method Aggregate of ChartController may be used. That method has the following arguments:

RawData – This is an array of the source data for aggregation. It should be 2-dim array of points (<date/time>, <numeric value>).

AggrData – This is an output array of aggregates. It is 2-dim array of points (<interval ID>, <aggregated value>). Here <interval ID> may be either 0-based interval index or date/time of the interval's starting point. This depends upon the value of the last argument of Aggregate.

AggrInterval – This variable is duration of the intervals for data aggregation (per second, minute, hour, day, month, and year). The possible values are:

a. Const dpSec = 0

b. Const dpMin = 1

c. Const dpHour = 2

d. Const dpDay = 3

e. Const dpMonth = 4

f. Const dpYear = 5

AggrType – This variable is the type of aggregated values (simple sum, weighted sum, simple average, weighted average, and minimum, maximum). The possible values are:

a. Const aggrSimpleSum = 0

b. Const aggrWeightedSum = 1

c. Const aggrSimpleAvg = 2

d. Const aggrWeightedAvg = 3

e. Const aggrMin = 4

f. Const aggrMax = 5

UseDateTimeAtAxisX – If this argument is True then <interval id> of the result is Date/Time, otherwise it is interval index.

Example 5 – Using Aggregate Function

Let us consider the following input time sequence.

Time	9:13:30	9:13:40	9:13:50	9:14:00	9:14:10	9:14:18	9:14:32	9:14:40	9:14:50	9:15:00
Value	0	1	2	3	4	5	6	7	8	9

Time	9:15:10	9:15:20	9:15:30	9:15:40	9:15:50	9:16:00	9:16:10	9:16:20	9:16:30	9:16:40
Value	10	11	12	13	14	15	16	17	18	19

The result of aggregation over minute depending upon aggregation type will be as follows (the first column show 2 possible values of the <interval id>).

Simple Sum

Interval ID	Value
0 / 9:13:00	0 + 1 + 2
1 / 9:14:00	3 + 4 + 5 + 6 + 7 + 8
2 / 9:15:00	9 + 10 + 11+ 12+ 13+ 14
3 / 9:16:00	15 + 16+ 17+ 18+ 19

Weighted Sum

Interval ID	Value
0 / 9:14:00	3 * 10 sec + 4 * 8 sec + 5 * 14 sec + 6 * 10 sec + 7 * 10 sec + 8 * 10 sec
1 / 9:15:00	9 * 10 sec + 10 * 10 sec + 11 * 10 sec + 12 * 10 sec + 13 * 10 sec + 14 * 10 sec

As you can see here we have just 2 intervals because there are insufficient data for the first and last minute of the initial data to calculate the weighted sum.

The value is multiplied by the time in seconds when this value holds. It is always 10 seconds in our example except the red points like shown in the table with the source data.

Simple Average

Interval ID	Value
0 / 9:13:00	(0 + 1 + 2) / 3
1 / 9:14:00	(3 + 4 + 5 + 6 + 7 + 8) / 6
2 / 9:15:00	(9 + 10 + 11+ 12+ 13+ 14) / 6
3 / 9:16:00	(15 + 16+ 17+ 18+ 19) / 5

Weighted Average

Interval ID	Value
0 / 9:14:00	(3 * 10 sec + 4 * 8 sec + 5 * 14 sec + 6 * 10 sec + 7 * 10 sec + 8 * 10 sec) / 60 sec
1 / 9:15:00	(9 * 10 sec + 10 * 10 sec + 11 * 10 sec + 12 * 10 sec + 13 * 10 sec + 14 * 10 sec) / 60 sec

Minimum

Interval ID	Value
0 / 9:13:00	0
1 / 9:14:00	3
2 / 9:15:00	9
3 / 9:16:00	15

Maximum

Interval ID	Value
0 / 9:13:00	2
1 / 9:14:00	8
2 / 9:15:00	14
3 / 9:16:00	19

Creating Several Series

Object ChartController has methods GetSeriesCount and GetSeriesParams that provide a universal approach to filling several series in the script.

- The first parameter is considered as having value equal to comma-separated list of values each corresponding to separate series.
- The script splits that value list into separate values and fills series for data obtained for each value. For instance, if the value is “tag1, tag2, tag3”, then 3 series are created for “tag1”, “tag2”, “tag3”, correspondingly.

Example 6 – Using GetSeriesCount and GetSeriesParams functions

In this example GetSeriesParas returns 3, and GetSeriesParams returns arrays

{"tag1", params (1), params (2)},
{"tag2", params (1), params (2)},
{"tag3", params (1), params (2)}
for corresponding series.

```
Function BuildReport(params)
   ReDim params (2)
   params (0) = "tag1,tag2,tag3"
   params (1) = DateSerial (2011,1,1) +TimeSerial (0,0,0)
   params (2) = DateSerial (2011,1,1) +TimeSerial (0,0,10)
   SQL = "SELECT DateTimeStamp, HistoryValue " & vbNewLine _
            & "FROM History " & vbNewLine _
            & "WHERE Tag = @TagName~ " & vbNewLine _
            & "   AND DateTimeStamp >= @DateFrom~ AND DateTimeStamp <= @DateTo~ " & vbNewLine _
            & "ORDER BY DateTimeStamp"
   With ChartController
   cntSeries = .GetSeriesCount(params)
   For i = 0 To cntSeries - 1
   arrSeriesParams = .GetSeriesParams(params, i)
            .RetrieveData(SQL, arrSeriesParams, arrRawData, False)
            .FillSeries i, -1, arrRawData
            .SetSeriesLegendTitle i, arrSeriesParams(0)
   Next
   SetReportCaption "Chart for period from " & params (1) & " to " & params (2)
   End With
End Function
```

Setting Series Title Displayed in the Legend

To set the series title display the method SetSeriesLegendTitle of ChartController object is used. It has the following arguments:

- **Index** – Series index, must point at existing series
- **LegendTitle** – Text that will be shown in the legend for the considered series.

Setting Report Caption

Set Report Caption by calling the method SetReportCaption of ChartController object.

Checking Script, Applications, and Reporting Errors

ChartController object has methods HasError and GetErrorText for error checking. All other methods when called reset the error, and if during their execution an error occurs corresponding errors are set.

To check and report errors when using ChartController in the script one may call HasErrors after each function call and if it returns True then report error returned by GetErrorText to a user specified location.

Example 7 – Using HasError and GetErrorText functions

```
Function BuildReport (params)
   ReDim arrChartData (9, 2)
   For i = 0 To 9
   arrChartData(i,0) = i
   arrChartData (i,1) = i
   Next
   ChartController.FillSeries 0, -1, arrChartData
   if ChartController.HasError then MsgBox ChartController.GetErrorText
End Function
```

Using Script Editor

General Functionality

In order to write the script the Script Editor must be opened. This Script Editor is open using property “Data Source” on “Chart Properties” panel. Double clicking that property brings up the Script Editor as shown on Figure #14.8.

- The script may be edited directly.
- The syntax of the script may be checked for errors any time by clicking **Check Syntax**.
- Clicking **Save Script** saves the script to the report book. Note that this does not save the script in the report book file, for this the report book itself should be saved.
- Clicking **Refresh Chart** calls the script function BuildReport passing default parameter values if function GetParamDefs exists in the script. If GetParamDefs function is missing or has errors then a message box will be shown where the user may choose between cancelling script run and running the script anyway. In the latter case an empty variant will be passed to the BuildReport function for *params* argument. This makes sense to be done if the script does not depend upon parameter values passed to the BuildReport function.
- Clicking **Wizard** launches the Script Wizard shown on Figure #14.9 below.

Using Script Wizard

Script Wizard assists the user with creating the script via generating skeleton code for typical situations. The steps are as follows:

1. Create a new report book.
2. Open **Script Editor** Dialog box.
3. Click **Wizard**. This will bring up the Script Wizard. See Figure #14.9.

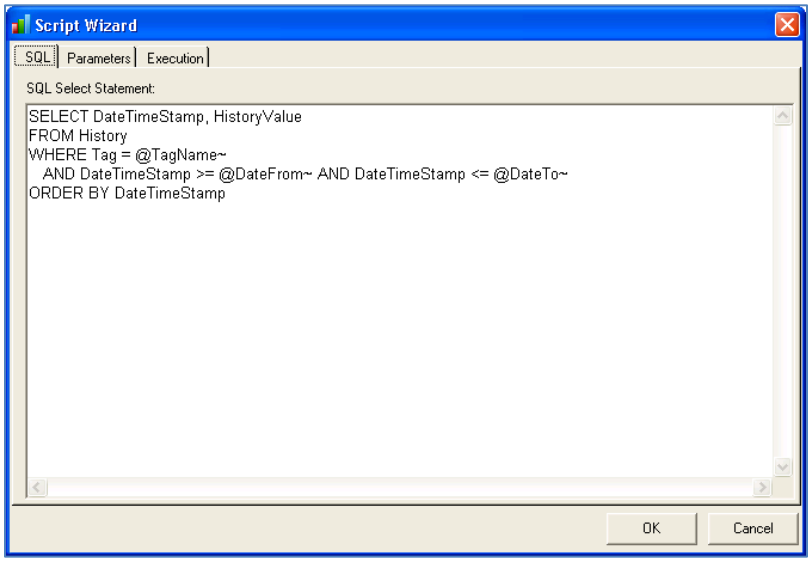


Figure #14.9 – Script Wizard

The Script Wizard has three-tab pages for defining report settings that are used for code generation.

- On the **SQL** tab, define SQL Select query with parameters. The default SQL query depends upon chart type – for Pie it selects average values of several tags for given time range, for the rest of chart types it selects all values for given tag and time range. Use the default query or create your own query.
- On **Parameters** tab define the properties for the report parameters. The list of the parameters is obtained from SQL query when **Refresh from SQL** is pressed. Initially parameters have order of their appearance in the query. Then the parameters may be reordered using GUI. The order of the parameters is important because the first parameter may be different as described in the **Accessing Database Data** section above (see note). Also, parameters are shown in the Report Editor Runtime form just in this order.

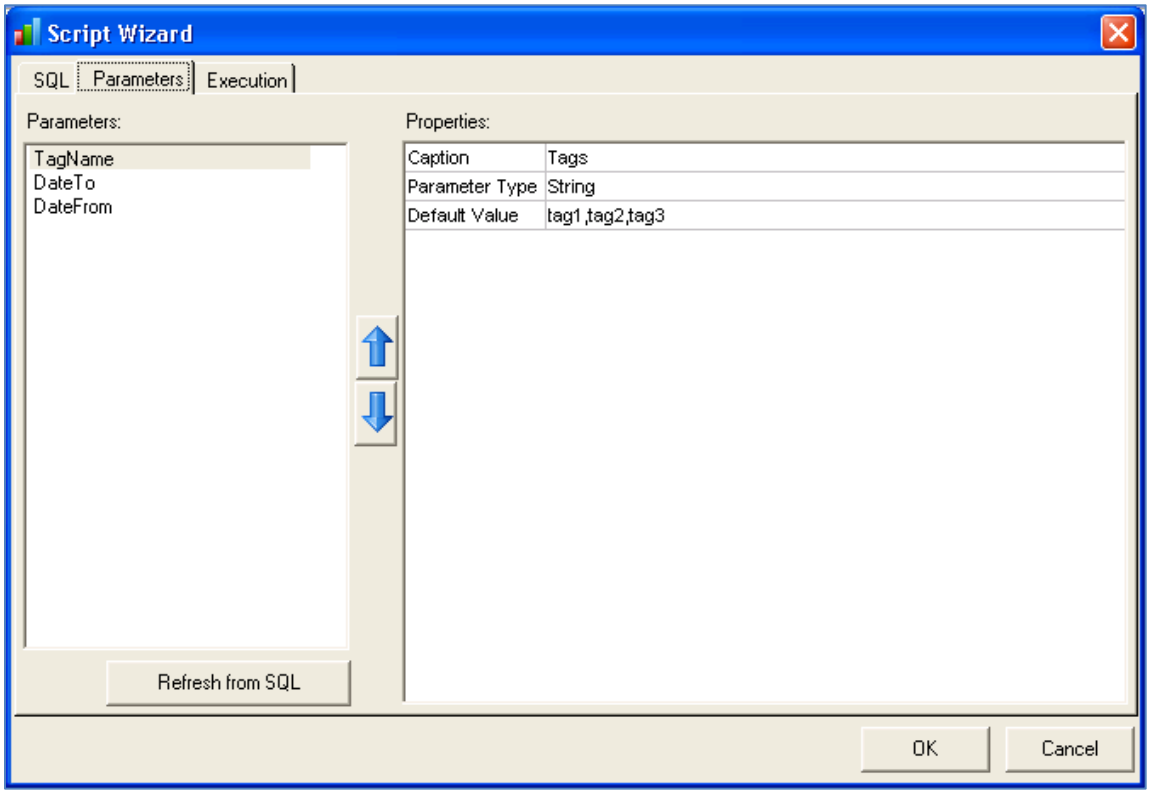


Figure #14.10 – Script Wizard Parameters Tab

- The **Execution** tab provides information if data aggregation should be done before showing the data in the chart. If "Show Raw Data" is selected then the data returned by the SQL query will be shown in the chart, otherwise corresponding data aggregation will be done.

  If chart type is Pie, the Execution tab is not present.

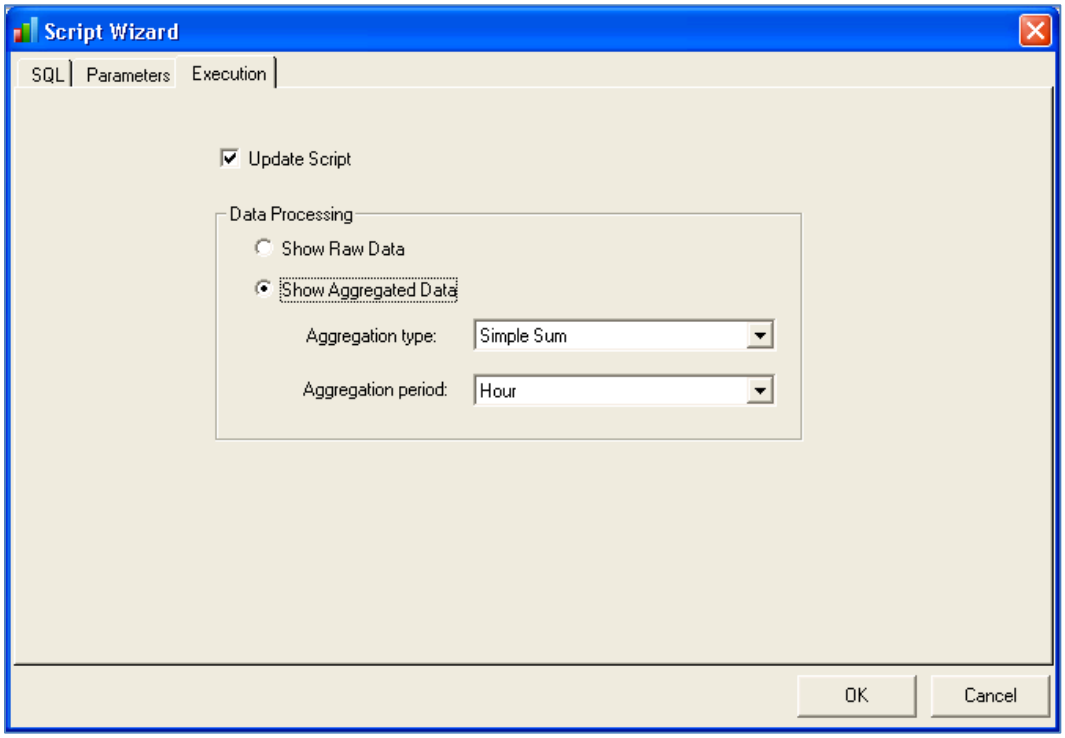


Figure #14.11 – Script Wizard Execution Tab

Press **OK** and the Wizard will generate ready-made script for filling the chart with data.

When the Wizard is next opened, it fills its pages SQL and Params via executing script functions SQL and GetParamsDef, correspondingly. If script functions were edited manually then corresponding changes will be reflected in the Wizard. That is not the same for Execution. It cannot be reconstructed from the BuildReport function. Checking **Update Script** makes it possible to prevent from overwriting BuildReport function by the Wizard. To prevent Script Wizard from overwriting manual changes to Execution the **Update Script** check box should be unchecked.

Using GraphRepCtrl Control

GraphRepControl is implemented in PrjGraphRepControl.ocx. It is designed for use in any ActiveX controls container, in ClearView Client, in particular. Applied to ClearView Client the control should be used the following way.

1. Register control in ClearView Client like any other ocx controls.
2. Place control on the screen.
3. Write the script that does the following:
    a. Set path to report book (property ReportPath).
    b. Set name of the report from the report book (property ReportName).
    c. Set connection string (property ConnectString).
    d. In order to see the report on screen call method Refresh passing comma-separated report parameter values.
    e. In order to preview and then print the report like it is shown in the chart call method PreviewChart.
    f. In order to print the report call method PrintChart passing comma-separated report parameter values; in contrast to PreviewChart that shows chart like shown on screen this method prints chart immediately to the printer using maximum advantage of the printer resolution.

g. The functions are executed asynchronously. For instance, Refresh returns immediately, while chart contents are changed after refresh. When operation is completed an event OperationFinished is raised. During execution of the current operation (refresh, print, or print preview) the functions Refresh, PrintChart, and PreviewChart do nothing immediately returning False until the operation is finished.

Example 8 – Example Screen using GraphRepControl

In this example the screen is designed showing tag values for tags tag1, tag2, and tag3 for a given day and given time range within that day.

It is assumed that there is a designed report in **C:\Program Files\ClearView\Bin\GraphReps.grb.xml**, the report name is SimpleTagValues, and the report parameters are <tags list>, <initial date>, <final date>.

The screen is shown in Figure #14.12. It has 3 controls where the user may set report date and report time range and then perform required operation.

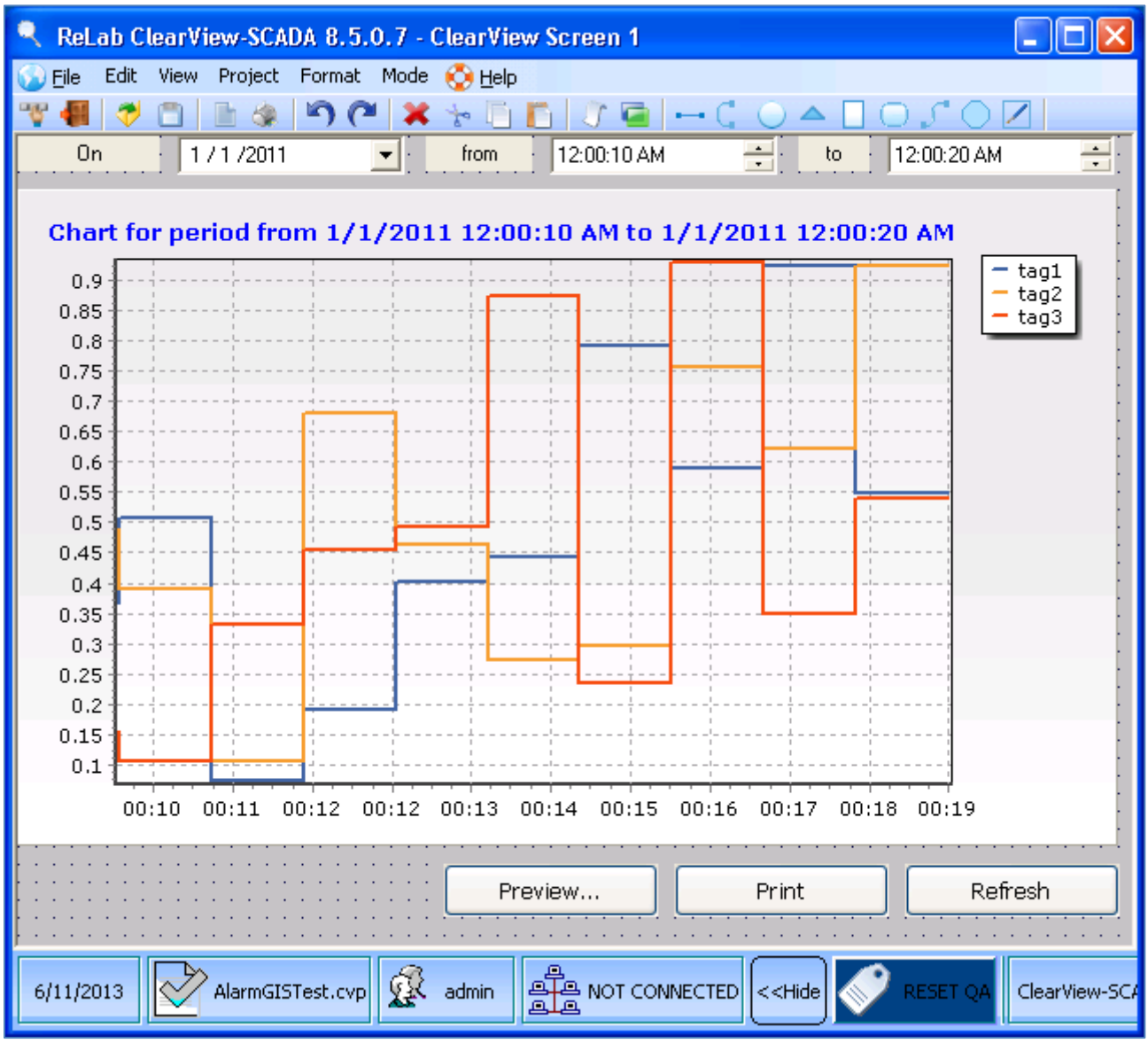


Figure #14.12 – Example of GraphRepControl

Screen implementation in the script is as follows:

```
Dim Previewing
ctlDate.Value = DateSerial (2011,1,1)
ctlTimeFrom.Format = 0
ctlTimeTo.Format = 0
ctlTimeFrom.Format = 2
ctlTimeTo.Format = 2
ctlTimeFrom.Value = TimeSerial (0,0,10)
ctlTimeTo.Value = TimeSerial (0,0,20)

‘ There are two ways to specify connection parameters, use one of them:
‘  1 – to specify ConnectString Property where the user has to specify the ADO connection string
ctlChart.ConnectString = "Provider=MSDASQL.1; Data Source=DSN name; User ID= UserName; Password=Password"

‘ 2 – to call the SetConnectParams method
ctlChart.SetConnectParams  "DSN name", "UserName", "Password"

ctlChart.ReportPath = "C:\Program Files\ClearView\Bin\GraphReps.grb.xml"
ctlChart.ReportName = "SimpleTagValues"
Previewing = False

Private Function GetDateFrom()
GetDateFrom = ctlDate.Value _
        + TimeSerial(Hour (ctlTimeFrom.Value), Minute(ctlTimeFrom.Value), Second(ctlTimeFrom.Value))
End Function

Private Function GetDateTo()
GetDateTo = ctlDate.Value _
        + TimeSerial (Hour (ctlTimeTo.Value), Minute (ctlTimeTo.Value), Second (ctlTimeTo.Value))
End Function

Private Sub btnRefresh_Click( )
ctlChart.Refresh "tag1, tag2, tag3", GetDateFrom, GetDateTo
End Sub

Private Sub btnPrint_Click( )
ctlChart.PrintChart "tag1, tag2, tag3", GetDateFrom, GetDateTo
End Sub

Private Sub btnPreview_Click( )
Previewing = True
ctlChart.Refresh "tag1, tag2, tag3", GetDateFrom, GetDateTo
End Sub

Private Sub ctlChart_OperationFinished( FinishedOper )
If FinishedOper = 0 And Previewing = True Then
tmrOperFinished.Enabled = True
```

```
End If
End Sub

Private Sub tmrOperFinished_Timer( )
tmrOperFinished.Enabled = False
ctlChart.PreviewChart
Previewing = False
End Sub
```

Helper functions GetDateFrom and GetDateTo are used for converting values from the controls into the values required by the report.

PrintPreview functionality. The function is not called immediately, instead Refresh is called first, and PrintPreview is called after Refresh is completed. PrintPreview is called in response to the OperationFinished event, completion of Refresh. PrintPreview cannot be called in this event handler because at this time Refresh operation is not completely finished.

SECTION 15

Historical Alarm and Events Control

Executable: CVDataControls.dll

Type Library: CVDATACONTROLSLib

Control: HistoryViewCtrl

Active View

Control shows historical data in 4 views.

View	Value	Enumeration
Alarms History	0	cvViewAlarms
Events	1	cvViewEvents
Security	2	cvViewSecurity
All	3	cvViewAll

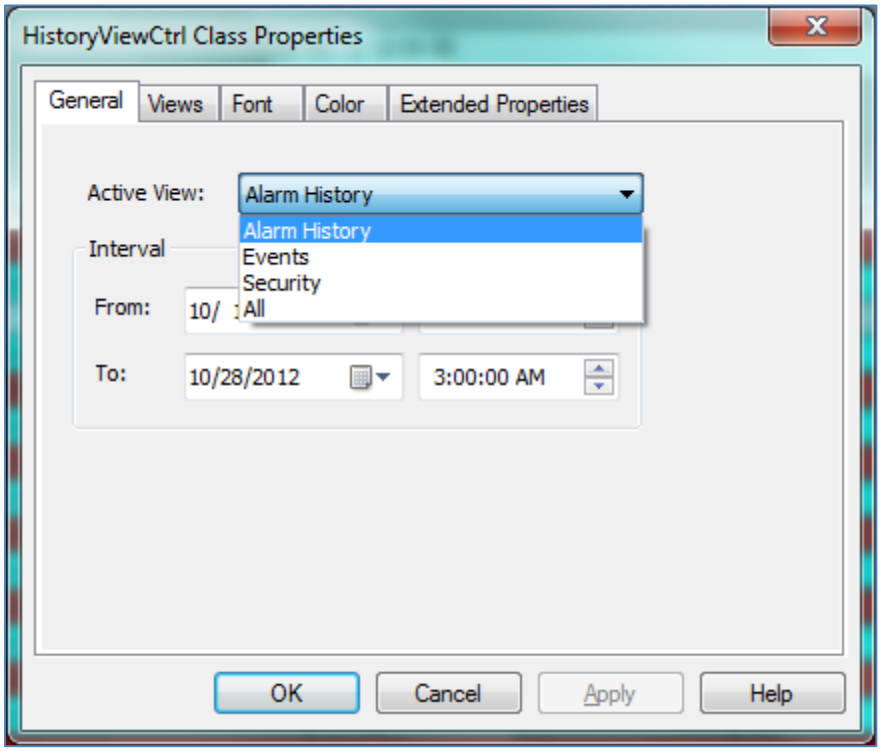


Figure #15.1

The views are activated at runtime using property ActiveView.

Control shows data from the database that is defined using SetConnectParams method. The control does not keep a permanent connection to the database; the database is connected only at the time of data retrieval.

Example:

```
Sub RefreshData ()
historyViewCtrl.SetConnectParams “ClearView”, “sa”, “password”
historyViewCtrl.ActiveView = cvViewSecurity
historyViewCtrl.Refresh
End sub
```

In case of an error in the Refresh method (usually this is due to missing access to the database), the control keeps showing data that it had before Refresh called.

Default Time Range Property

The time interval is used to select data from the database while an additional filter is applied to show just those data retrieved from database that meets some extra criteria.

Example:

```
historyViewCtrl.DateFrom = “20111003 183045000”
historyViewCtrl.DateTo = “20111010 183045000”
```

Note: Date format is: YYYYMMDD hhmmssnnn (data type “String”).

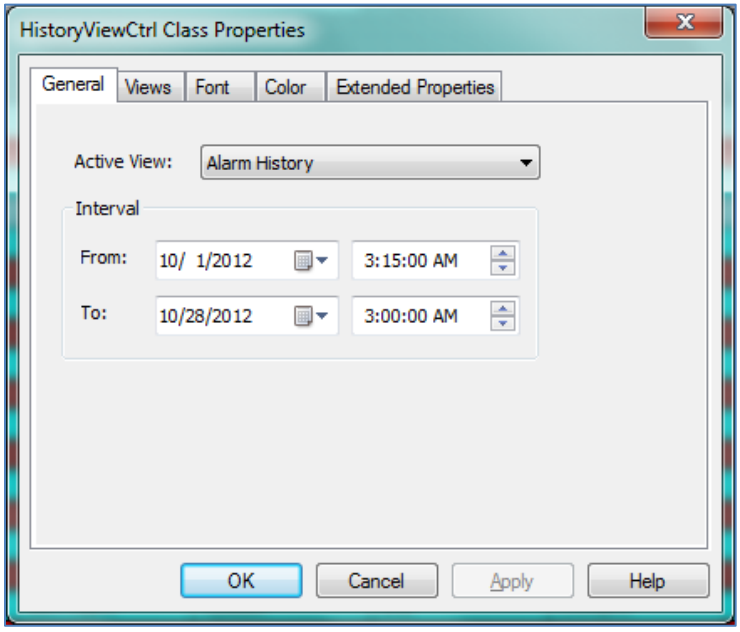


Figure #15.2

Filtering

Time Range Filtering Method

The range time interval is used to query the database for a specific time range.

Time Range Example (time query select):

historyViewCtrl.SetTimeRange CDate(TimeFrom), CDate(TimeTo)

historyViewCtrl.Refresh

Column Filtering Property

Column filtering consists of FilterColumn, FilterOper, FilterValue, and FilterColumn_XXX, FilterOper_XXX, FilterValue_XXX. The first 3 properties are used to define a filter for the cvViewAlarms view; the others define other views. (Here *XXX* stands for the name of the corresponding view.)

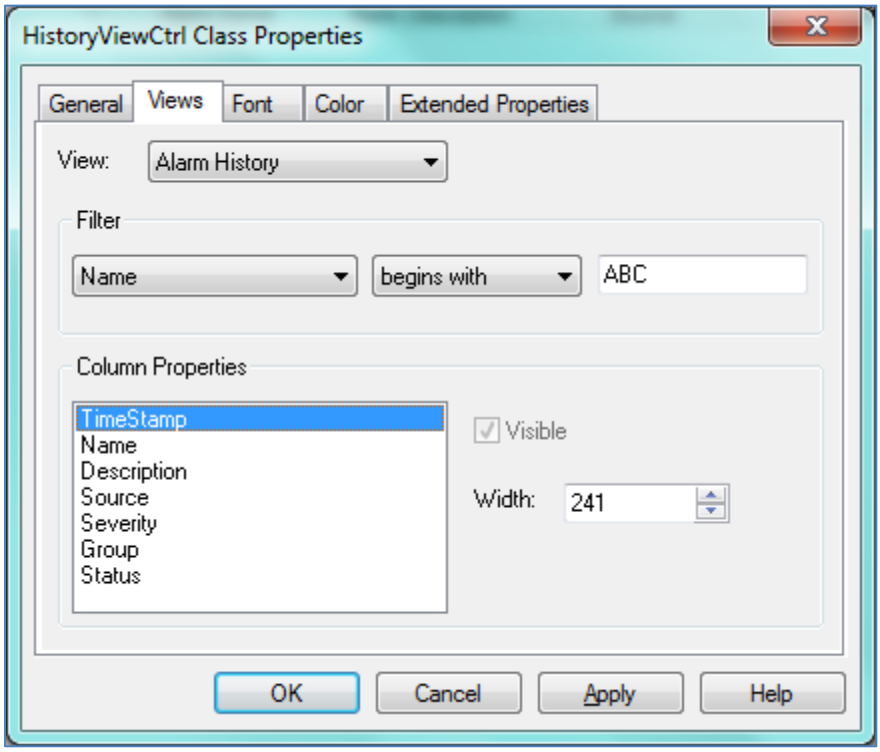


Figure #15.3

Alarm History Column filtering

FilterColumn (property)

Column Name	Value	Enumeration
Name	1	cvColAlarmName
Description	2	cvColAlarmDescription

Column Name	Value	Enumeration
Source	3	cvColAlarmSource
Severity	4	cvColAlarmSeverity
Group	5	cvColAlarmGroup
Status	6	cvColAlarmStatus

Events Column filtering

FilterColumn_Events (property)

Column Name	Value	Enumeration
Source	1	cvColSource
Severity	2	cvColSeverity
Message	3	cvColMessage

Security Column filtering

FilterColumn_*Security* (property)

Column Name	Value	Enumeration
Source	1	cvColSource
Severity	2	cvColSeverity
Message	3	cvColMessage

All Column filtering

FilterColumn_*All* (property)

Column Name	Value	Enumeration
Source	1	cvColSource
Severity	2	cvColSeverity
Message	3	cvColMessage

Alarm History filtering operators

FilterOper (property)

Operator	Value	Enumeration
No Filter	0	cvCompNoFilter
Equal	1	cvCompEqual
No Equal	2	cvCompNotEqual
Greater	3	cvCompGreater
Not Greater	4	cvCompNotGreater
Less	5	cvCompLess
Not Less	6	cvCompNotLess
Begins With	7	cvCompBeginsWith
Includes	8	cvCompIncludes
Is One Of	9	cvCompIsOneOf

Events filtering operators

FilterOper_Events (property)

Operator	Value	Enumeration
No Filter	0	cvCompNoFilter
Equal	1	cvCompEqual
No Equal	2	cvCompNotEqual
Greater	3	cvCompGreater
Not Greater	4	cvCompNotGreater
Less	5	cvCompLess
Not Less	6	cvCompNotLess
Begins With	7	cvCompBeginsWith
Includes	8	cvCompIncludes
Is One Of	9	cvCompIsOneOf

Security filtering operators

FilterOper_*Security* (property)

Operator	Value	Enumeration
No Filter	0	cvCompNoFilter
Equal	1	cvCompEqual
No Equal	2	cvCompNotEqual
Greater	3	cvCompGreater
Not Greater	4	cvCompNotGreater
Less	5	cvCompLess
Not Less	6	cvCompNotLess
Begins With	7	cvCompBeginsWith
Includes	8	cvCompIncludes
Is One Of	9	cvCompIsOneOf

All filtering operator

FilterOper_All (property)

Operator	Value	Enumeration
No Filter	0	cvCompNoFilter
Equal	1	cvCompEqual
No Equal	2	cvCompNotEqual
Greater	3	cvCompGreater
Not Greater	4	cvCompNotGreater
Less	5	cvCompLess
Not Less	6	cvCompNotLess
Begins With	7	cvCompBeginsWith
Includes	8	cvCompIncludes

Operator	Value	Enumeration
Is One Of	9	cvCompIsOneOf

Example (applying filter):

historyViewCtrl.FilterColumn = 1

historyViewCtrl.FilterOper = 7

historyViewCtrl.FilterValue = “ABC”

historyViewCtrl.Refresh

Example (removing filter):

historyViewCtrl.FilterOper = 0

historyViewCtrl.Refresh

Resizing Columns

To resize columns programmatically, use the ColumnWidthXXX properties. XXX identifies the column and is the name of corresponding column on cvViewAlarms view or column name plus suffix identifying view for other views.

- View Alarm History [Width Property]
  - ColumnWidthTimestamp
  - ColumnWidthAlarmName
  - ColumnWidthAlarmDescr
  - ColumnWidthSource
  - ColumnWidthSeverity
  - ColumnWidthGroups
  - ColumnWidthStatus
- View Events [Width Property]
  - ColumnWidthTimestamp_Events
  - ColumnWidthSource_Events
  - ColumnWidthSeverity_Events
  - ColumnWidthMessage_Events
- View Security [Width Property]
  - ColumnWidthTimestamp_Security
  - ColumnWidthSource_Security

- ColumnWidthSeverity_Security
- ColumnWidthMessage_Security

- View All [Width Property]
  - ColumnWidthTimestamp_All
  - ColumnWidthSource_All
  - ColumnWidthSeverity_All
  - ColumnWidthMessage_All

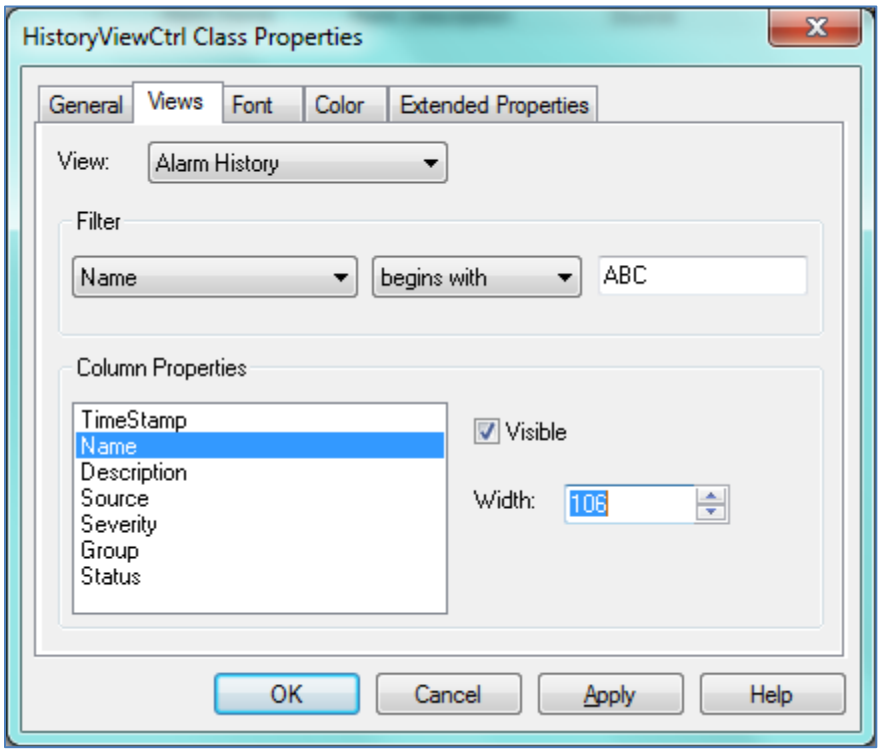


Figure #15.4

Example (resizing AlarmName column on alarms view)
historyViewCtrl.ColumnWidthAlarmName = 106

Example (resizing Severity column on All view)
historyViewCtrl. ColumnWidthSeverity_All = 300

An alternative way is to have the ColumnWidth property use the ColumnIndex parameter. This method applies to the current view.

Example (resizing AlarmName column on alarms view)
historyViewCtrl.ColumnWidth(1) = 300

Columns Visible/Invisible

To resize columns programmatically, use the ColumnVisibleXXX properties. XXX identifies the column and is the name of corresponding column on cvViewAlarms view or column name plus suffix identifying view for other views.

- View Alarm History [Visible Property]
  - ColumnVisibleTimestamp
  - ColumnVisibleAlarmName
  - ColumnVisibleAlarmDescr
  - ColumnVisibleSource
  - ColumnVisibleSeverity
  - ColumnVisibleGroups
  - ColumnVisibleStatus
- View Events [Visible Property]
  - ColumnVisibleTimestamp_Events
  - ColumnVisibleSource_Events
  - ColumnVisibleSeverity_Events
  - ColumnVisibleMessage_Events
- View Security [Visible Property]
  - ColumnVisibleTimestamp_Security
  - ColumnVisibleSource_Security
  - ColumnVisibleSeverity_Security
  - ColumnVisibleMessage_Security
- View All [Visible Property]
  - ColumnVisibleTimestamp_All
  - ColumnVisibleSource_All
  - ColumnVisibleSeverity_All
  - ColumnVisibleMessage_All

Example (hiding AlarmName column on alarms view)
historyViewCtrl. ColumnVisibleAlarmName = False

Example (hiding Severity column on All view)
historyViewCtrl. ColumnVisibleSeverity_All = False

An alternative way is to have the ColumnVisible property use the ColumnIndex parameter. This method applies to the current view.

Example (hiding AlarmName column on alarms view)
historyViewCtrl.ColumnVisible(1) = False

Changing Colors

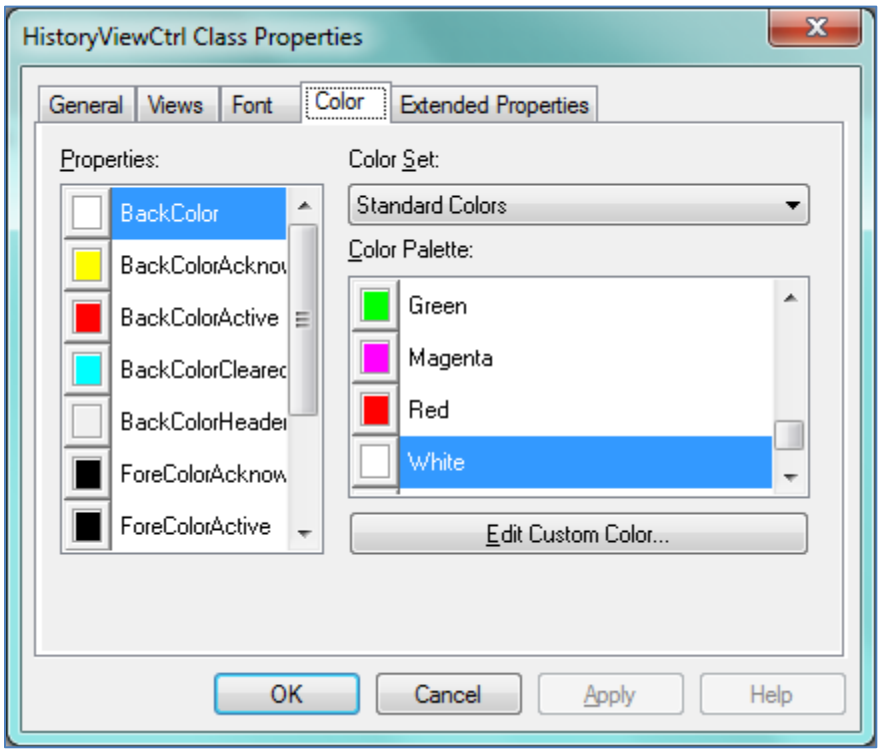


Figure #15.5

- Control Back Color (BackColor)
- Acknowledged Alarm Raw Color (BackColorAcknowledged)
- Active Alarm Raw Color (BackColorActive)
- Cleared Alarm Raw Color (BackColorCleared)
- Header Color (BackColorHeader)
- Acknowledged Font Color (ForeColorAcknowledged)
- Active Alarm Font Color (ForeColorActive)
- Cleared Alarm Font Color (ForeColorCleared)
- Header Font Color (ForeColorHeader)

The properties change the colors used when in Design and Run mode.

Example (changing the colors used to display cleared alarms):

historyViewCtrl. ForeColorCleared = 255

historyViewCtrl. BackColorCleared = vbWhite

Changing Fonts

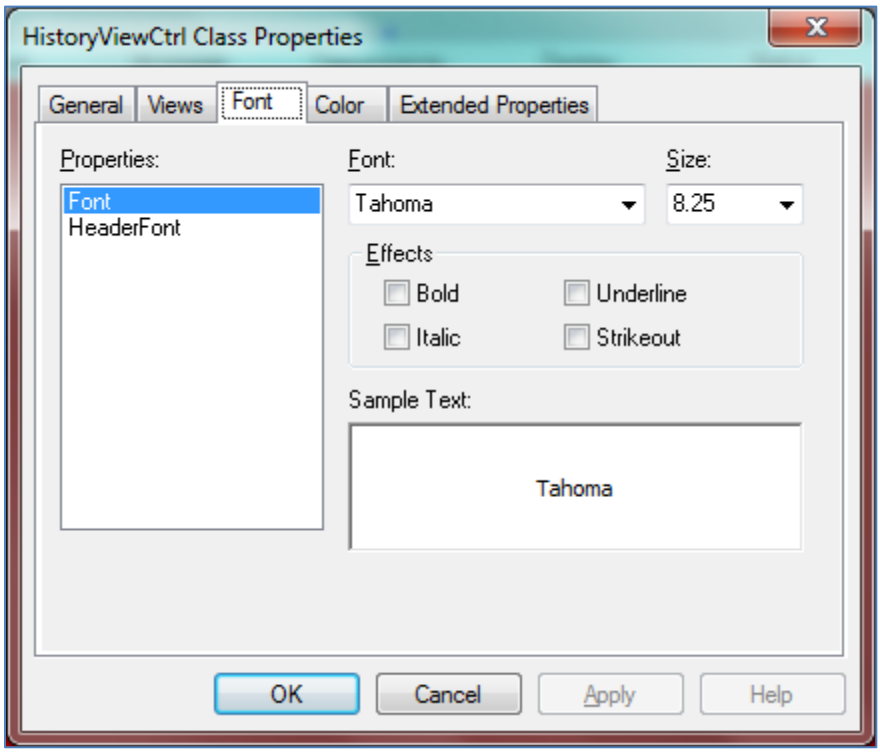


Figure #15.6

The properties are for changing colors when in Design and Run mode.

Font Property	HeaderFont Property
ctrlHistoryView.Font.Bold	ctrlHistoryView.HeaderFont.Bold
ctrlHistoryView.Font.Charset	ctrlHistoryView.HeaderFont.Charset
ctrlHistoryView.Font.Italic	ctrlHistoryView.HeaderFont.Italic
ctrlHistoryView.Font.Name	ctrlHistoryView.HeaderFont.Name
ctrlHistoryView.Font.Size	ctrlHistoryView.HeaderFont.Size
ctrlHistoryView.Font.Strikethrough	ctrlHistoryView.HeaderFont.Strikethrough
ctrlHistoryView.Font.Underline	ctrlHistoryView.HeaderFont.Underline

Methods reference

Name	Description	Example
SetConnectParams	Database Connection Parameters where “ClearView” is DSN “sa” is user name “password” is password	historyViewCtrl.SetConnectParams “ClearView”, “sa”, “password”
Refresh	Refresh method has two capabilities: Refreshing data (database query) if the time range changed Or refreshing data regardless if the time range changed or not. The default is “NO” database refresh.	No database Refresh if the time range did not change: ctrlHistoryView.Refresh or ctrlHistoryView.Refresh False Force Database refresh: ctrlHistoryView.Refresh True
SetTimeRange	Method specifies the query time range (From and To date and time).	ctrlHistoryView.SetTimeRange CDate(Now() - 1), CDate(Now() + 1)
AddSortCol	Data are sorted via clicking the control’s header. It is also possible to sort data programmatically. This way the data may be sorted by several columns.	historyViewCtrl. ClearSortCols historyViewCtrl. AddSortCol 1, cvSortAscending historyViewCtrl. AddSortCol 4, cvSortDescending historyViewCtrl. Refresh
ClearSortCols	Clears current sorting.	historyViewCtrl. ClearSortCols
SetCustomColors	Changing colors programmatically	historyViewCtrl. SetCustomColors cvAlarmCleared, VBRed, VBWhite
SetDisplayMode	Method displays event sequence when the particular raw is selected.	Applies filter: ctrlHistoryView.SetDisplayMode 1 Removes filter: ctrlHistoryView.SetDisplayMode 1

Event Sequence Interface

Right-click selected row, then select **Show Event Sequence** to view and set a sequence of events. Selecting **Show Normal** will remove the filter.

TimeStamp	Alarm Name	Alarm Description	Source	Severity	Groups	Status
10/08/2012 09:06:11.096	Sim1_Alarm	Sim1 Alarm	System	Warning	CVAlarm	Set
10/08/2012 09:06:00.114	Sim1_Alarm	Sim1 Alarm	System	Info	CVAlarm	Cleared
10/08/2012 09:05:19.258	Sim1_Alarm	Sim1 Alarm	Default_User A...	Info	CVAlarm	Ack
10/0		Sim1 Alarm	System	Warning	CVAlarm	Set
10/0		Sim1 Alarm	System	Warning	CVAlarm	Set
10/0		Sim1 Alarm	System	Info	CVAlarm	Cleared
10/05/2012 17:55:11.242	Sim1_Alarm	Sim1 Alarm	System	Warning	CVAlarm	Set
10/05/2012 17:55:00.087	Sim1_Alarm	Sim1 Alarm	System	Info	CVAlarm	Cleared
10/05/2012 17:54:11.133	Sim1_Alarm	Sim1 Alarm	System	Warning	CVAlarm	Set
10/05/2012 17:54:00.263	Sim1_Alarm	Sim1 Alarm	System	Info	CVAlarm	Cleared
10/05/2012 17:53:11.307	Sim1_Alarm	Sim1 Alarm	System	Warning	CVAlarm	Set
10/05/2012 17:53:00.152	Sim1_Alarm	Sim1 Alarm	System	Info	CVAlarm	Cleared
10/05/2012 17:52:11.480	Sim1_Alarm	Sim1 Alarm	System	Warning	CVAlarm	Set
10/05/2012 17:52:00.325	Sim1_Alarm	Sim1 Alarm	System	Info	CVAlarm	Cleared
10/05/2012 17:51:11.653	Sim1_Alarm	Sim1 Alarm	System	Warning	CVAlarm	Set
10/05/2012 17:51:00.500	Sim1_Alarm	Sim1 Alarm	System	Info	CVAlarm	Cleared
10/05/2012 17:50:11.096	Sim1_Alarm	Sim1 Alarm	System	Warning	CVAlarm	Set
10/05/2012 17:50:00.673	Sim1_Alarm	Sim1 Alarm	System	Info	CVAlarm	Cleared
10/05/2012 17:49:11.270	Sim1_Alarm	Sim1 Alarm	System	Warning	CVAlarm	Set
10/05/2012 17:49:00.116	Sim1_Alarm	Sim1 Alarm	System	Info	CVAlarm	Cleared

Show Normal
Show Event Sequence

Figure #15.7

TimeStamp	Alarm Name	Alarm Description	Source	Severity	Groups	Status
00/00/0000 00:00:00.000	Sim1_Alarm	Sim1 Alarm	System	Warning	CVAlarm	Set
00/00/0000 00:00:48.672	Sim1_Alarm	Sim1 Alarm	System	Info	CVAlarm	Cleared
00/00/0000 00:00:59.827	Sim1_Alarm	Sim1 Alarm	System	Warning	CVAlarm	Set
00/00/0000 00:01:48.783	Sim1_Alarm	Sim1 Alarm	System	Info	CVAlarm	Cleared
00/00/0000 00:01:59.653	Sim1_Alarm	Sim1 Alarm	System	Warning	CVAlarm	Set
00/00/0000 00:02:48.607	Sim1_Alarm	Sim1 Alarm	System	Info	CVAlarm	Cleared
00/00/0000 00:02:59.762	Sim1_Alarm	Sim1 Alarm	System	Warning	CVAlarm	Set
00/00/0000 00:03:49.165	Sim1_Alarm	Sim1 Alarm	System	Info	CVAlarm	Cleared
00/00/0000 00:03:59.601	Sim1_Alarm	Sim1 Alarm	System	Warning	CVAlarm	Set
00/02/0000 15:12:59.789	Sim1_Alarm	Sim1 Alarm	System	Warning	CVAlarm	Set
00/02/0000 15:13:07.778	Sim1_Alarm	Sim1 Alarm	Default_User A...	Info	CVAlarm	Ack
00/02/0000 15:13:48.634	Sim1_Alarm	Sim1 Alarm	System	Info	CVAlarm	Cleared
00/02/0000 15:13:59.616	Sim1_Alarm	Sim1 Alarm	System	Warning	CVAlarm	Set

Figure #15.8

SECTION 16

ClearView Historian Control

Executable: CVDataControls.dll

Type Library: CVDATACONTROLSLib

Control: TagHistoryCtrl

Active View

Control shows historical data in 4 views.

View	Value	Enumeration
Simple	0	cvViewTagValues
Aggregates	1	cvViewTagAggregates
Maximums	2	cvViewTagMax
Minimums	3	cvViewTagMin

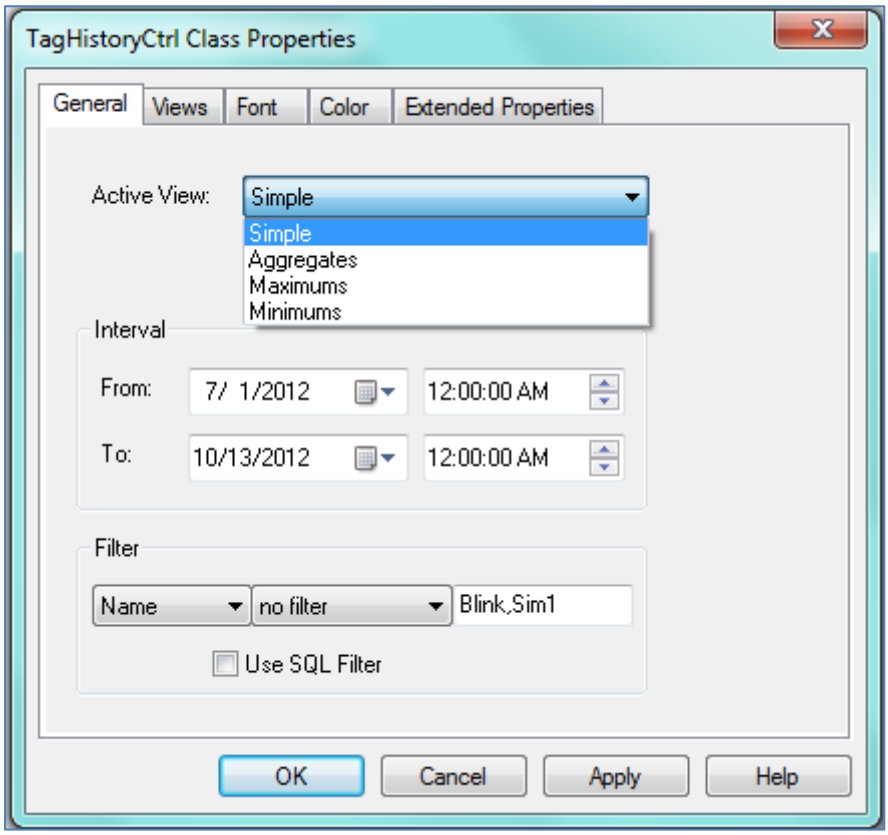


Figure #16.1

The views are activated at runtime using the ActiveView property.

Control shows data from the database that is defined using the SetConnectParams method. The control does not keep a permanent connection to the database; the database is connected only at the time of data retrieval.

Example:

```
Sub RefreshData()
TagHistoryCtrl.SetConnectParams “ClearView”, “sa”, “password”
TagHistoryCtrl.ActiveView 0
TagHistoryCtrl.Refresh
End sub
```

In case of an error in the Refresh method (usually this is due to missing access to the database), the control keeps showing data that it had before Refresh was called.

Default Time Range Property

The time interval is used to select data from the database while an additional filter is applied to show just that data retrieved from the database that meets some extra criteria.

Example:

```
TagHistoryCtrl.DateFrom = “20111003 183045000”
TagHistoryCtrl.DateTo = “20111010 183045000”
```

Note: Date format is: YYYYMMDD hhmmssnnn (data type “String”).

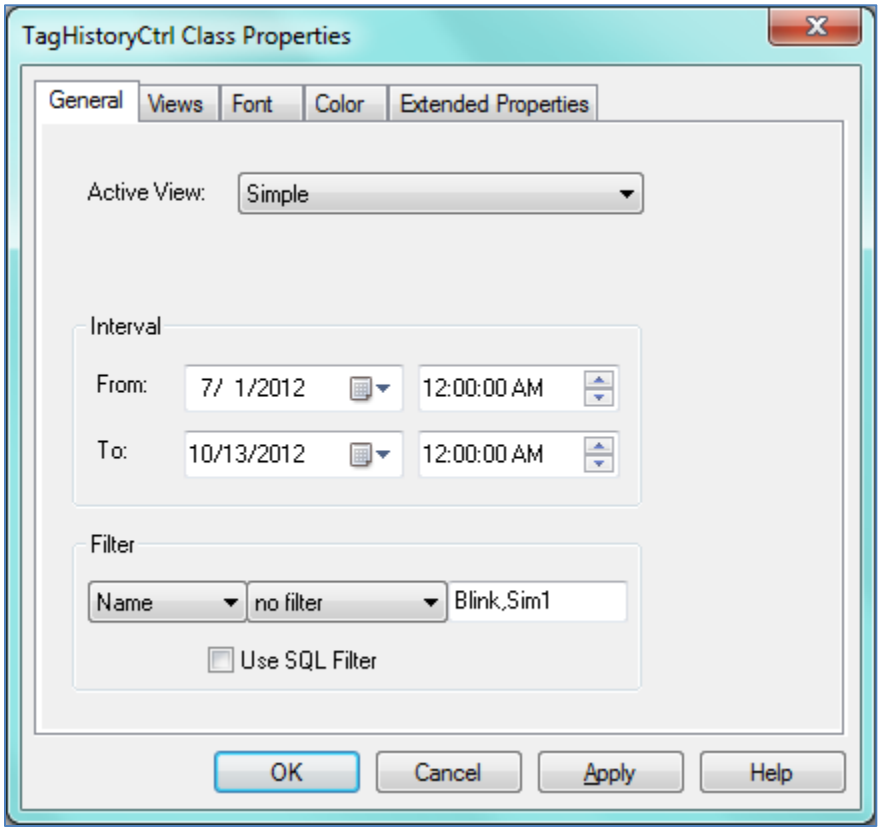


Figure #16.2

Filtering

Time Range Filtering Method

The time range interval is used to query the database for a specific time range.

Time Range Example (time query select):

TagHistoryCtrl.SetTimeRange CDate(TimeFrom), CDate(TimeTo)

TagHistoryCtrl.Refresh

Column Filtering Property

Column filtering consists of FilterColumn, FilterOper, FilterValue, FilterColumn_XXX, FilterOper_XXX, and FilterValue_XXX. The first 3 properties are used to define a filter for a specific view; the others define other views. (Here XXX stands for the name of the corresponding view.)

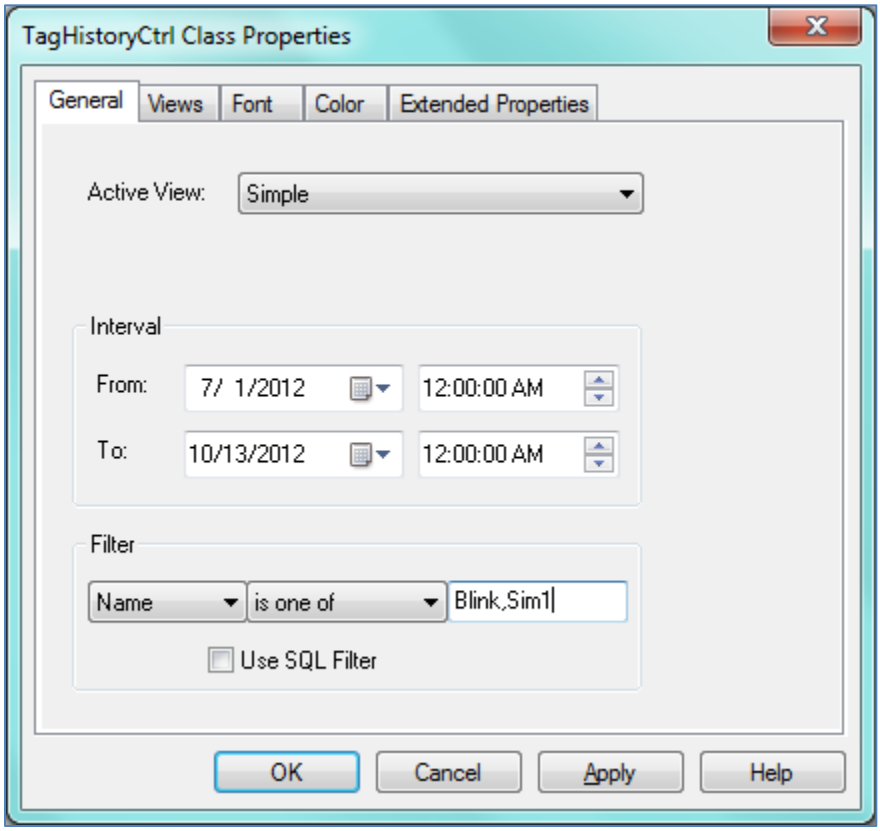


Figure #16.3

Filtered Columns

***TagFilterColumn* (property)**

Column Name	Value	Enumeration
Name	0	cvTagName
Decryption	1	cvTagDecr

Filtered Operators

TagFilterOper (property)

Operator	Value	Enumeration
No Filter	0	cvCompNoFilter
Equal	1	cvCompEqual
No Equal	2	cvCompNotEqual
Greater	3	cvCompGreater

Operator	Value	Enumeration
Not Greater	4	cvCompNotGreater
Less	5	cvCompLess
Not Less	6	cvCompNotLess
Begins With	7	cvCompBeginsWith
Includes	8	cvCompIncludes
Is One Of	9	cvCompIsOneOf

Example (applying filter):

TagHistoryCtrl.TagFilterColumn = 0

TagHistoryCtrl.TagFilterOper = 7

TagHistoryCtrl.TagFilterValue = “ABC”

TagHistoryCtrl.Refresh

Example (removing filter):

TagHistoryCtrl.TagFilterOper = 0

TagHistoryCtrl.Refresh

Aggregation

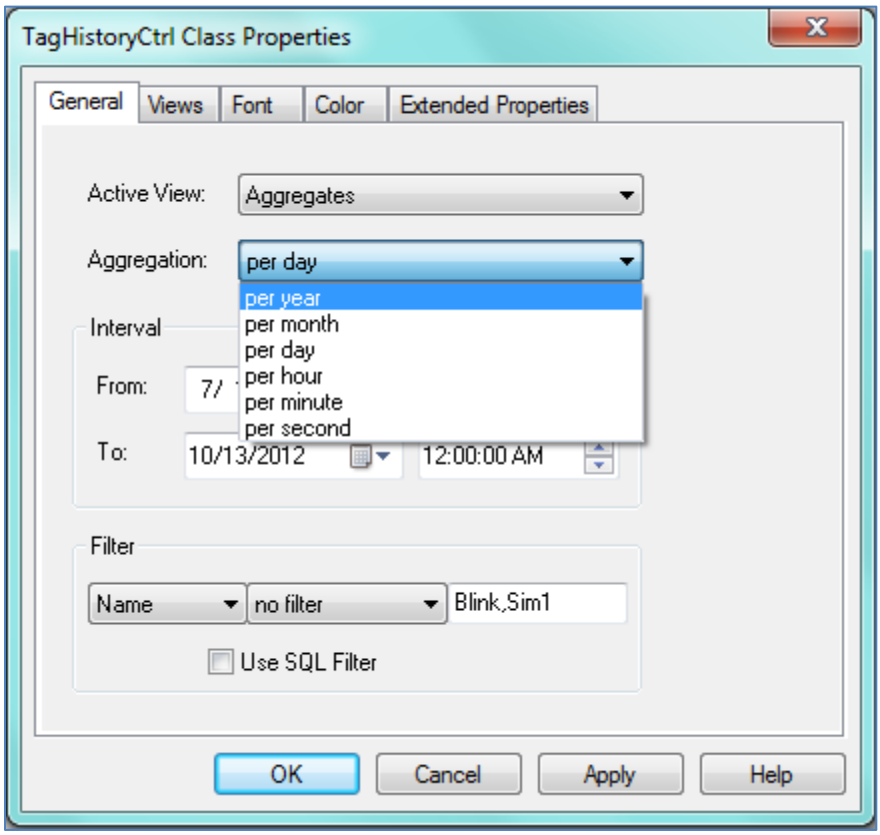


Figure #16.4

Time Aggregation

AggregationRange (Property)

Operator	Value	Enumeration
Per year	0	cvAggrStepYear
Per month	1	cvAggrStepMonth
Per day	2	cvAggrStepDay
Per hour	3	cvAggrStepHour
Per minute	4	cvAggrStepMin
Per second	5	cvAggrStepSec

Use SQL Filter (ServerFiltering) Property. Provides more efficient database querying.

Resizing Columns

To resize columns programmatically, use the ColumnWidthXXX property. XXX identifies the column and is the name of corresponding column name.

Column Description	Property
TimeStamp	ColumnWidthTimestamp
Tag Name	ColumnWidthTagName
Tag Description	ColumnWidthTagDesc
Tag Value	ColumnWidthTagValue
Tag Max Value	ColumnWidthTagMax
Tag Min Value	ColumnWidthTagMin
Tag Simple Average	ColumnWidthTagSimpleAvg
Tag Weighted Average	ColumnWidthTagWeightedAvg
Tag Sum	ColumnWidthTagSum

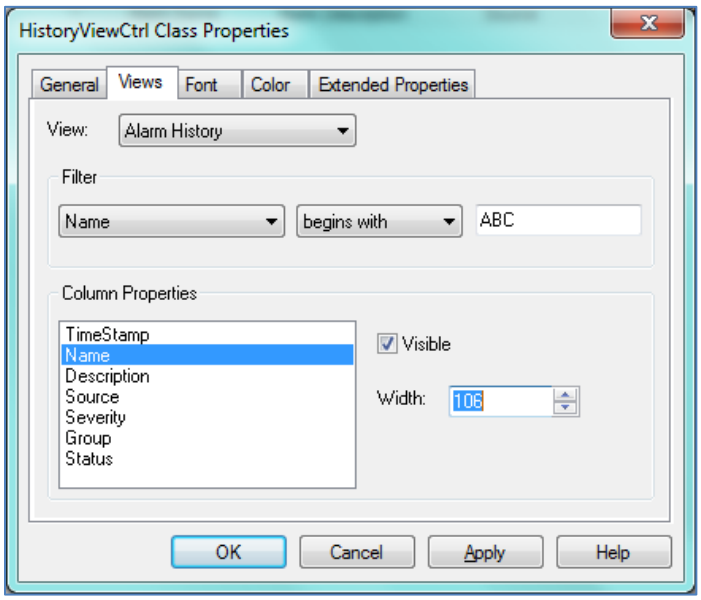


Figure #16.5

Example (resizing TagName column)

TagHistoryCtrl.ColumnWidthTagName = 100

Columns Visibility

To change columns visibility programmatically, use the ColumnVisibleXXX property. XXX identifies the column and is the name of the corresponding column.

Description	Property
TimeStamp	ColumnVisibleTimestamp
Tag Name	ColumnVisibleTagName
Tag Description	ColumnVisibleTagDesc
Tag Value	ColumnVisibleTagValue
Tag Max Value	ColumnVisibleTagMax
Tag Min Value	ColumnVisibleTagMin
Tag Simple Average	ColumnVisibleTagSimpleAvg
Tag Weighted Average	ColumnVisibleTagWeightedAvg
Tag Sum	ColumnVisibleTagSum

Example (hiding TagName column)

TagHistoryCtrl. ColumnVisibleTagName = False

Changing Colors

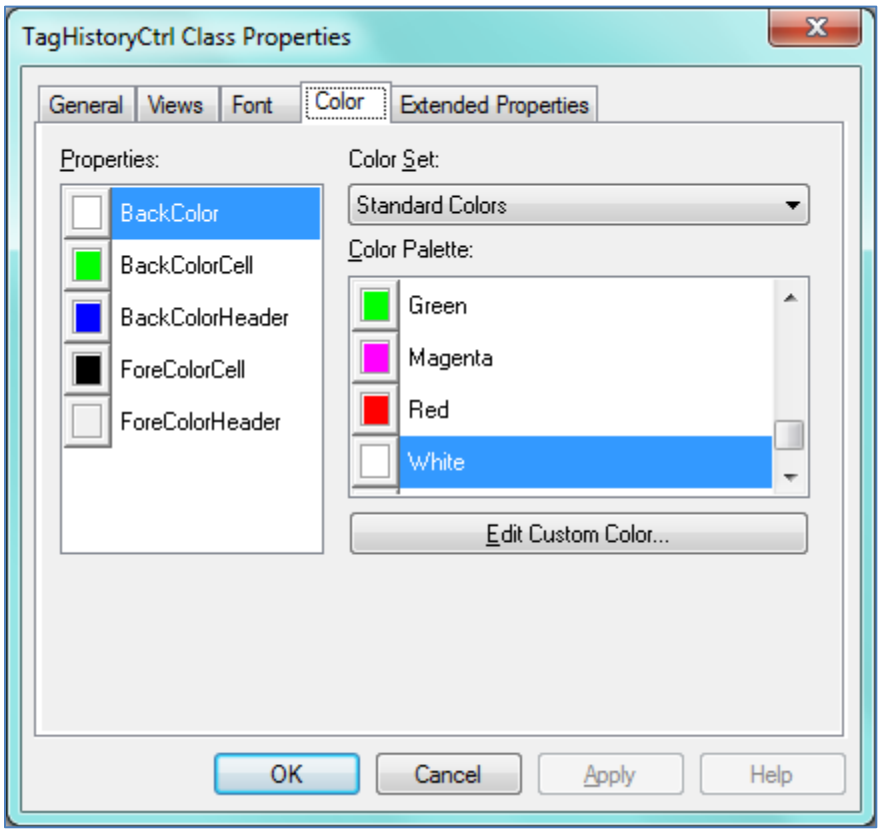


Figure #16.6

Property **BackColor** specifies Control's Back Color;

Property **BackColorCell** specifies Control's Cell Color;

Property **BackColorHeader** specifies Control's Header Color;

Property **ForeColorCell** specifies Control's Cell Font Color;

Property **ForeColorHeader** specifies Control's Header Font Color;

Changing Fonts

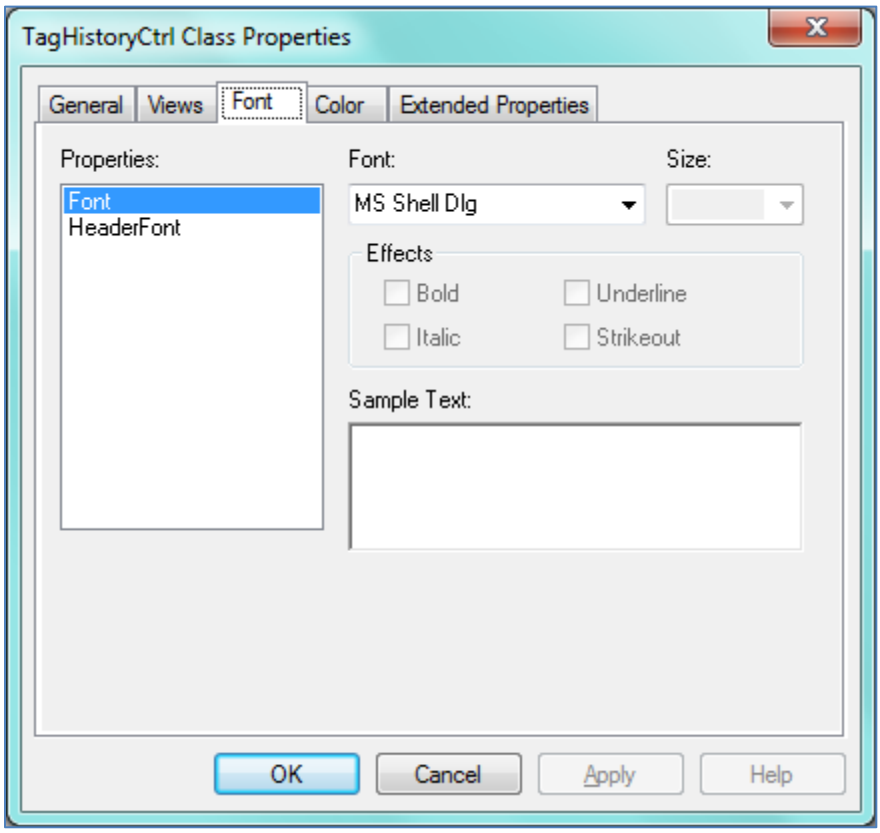


Figure #16.7

The properties are for changing the colors when in Design and Run mode.

Font Property	HeaderFont Property
TagHistoryCtrl.Font.Bold	TagHistoryCtrl.HeaderFont.Bold
TagHistoryCtrl.Font.Charset	TagHistoryCtrl.HeaderFont.Charset
TagHistoryCtrl.Font.Italic	TagHistoryCtrl.HeaderFont.Italic
TagHistoryCtrl.Font.Name	TagHistoryCtrl.HeaderFont.Name
TagHistoryCtrl.Font.Size	TagHistoryCtrl.HeaderFont.Size
TagHistoryCtrl.Font.Strikethrough	TagHistoryCtrl.HeaderFont.Strikethrough
TagHistoryCtrl.Font.Underline	TagHistoryCtrl.HeaderFont.Underline

Methods Reference

Name	Description	Example
SetConnectParams	Database Connection Parameters where "ClearView" is DSN "sa" is user name "password" is password	TagHistoryCtrl.SetConnectParams "ClearView", "sa", "password"
Refresh	Refresh method has two capabilities: Refreshing data (database query) if the time range changed Or refreshing data regardless if the time range changed or not. The default is "NO" database refresh.	No database Refresh if the time range did not change: TagHistoryCtrl.Refresh or TagHistoryCtrl.Refresh **False** Force Database refresh: TagHistoryCtrl.Refresh **True**
SetTimeRange	Method specifies the query time range (From and To date and time)	TagHistoryCtrl.SetTimeRange CDate(Now() - 1), CDate(Now() + 1)
AddSortCol	Data are sorted via clicking the control's header. It is also possible to sort data programmatically. This way the data may be sorted by several columns.	TagHistoryCtrl. ClearSortCols TagHistoryCtrl.AddSortCol 1, cvSortAscending TagHistoryCtrl.AddSortCol 4, cvSortDescending TagHistoryCtrl.Refresh
ClearSortCols	Clears current sorting.	TagHistoryCtrl.ClearSortCols
SetDisplayMode	Method displays event sequence when the particular row is selected	**Applies filter:** TagHistoryCtrl.SetDisplayMode 1 Removes filter: TagHistoryCtrl.SetDisplayMode 1

Event Sequence Interface

Right-click selected row, then select **Show Event Sequence** to view and set a sequence of events. Selecting **Show Normal** will remove the filter.

TimeStamp	Alarm Name	Alarm Description	Source	Severity	Groups	Status
10/08/2012 09:06:11.096	Sim1_Alarm	Sim1 Alarm	System	Warning	CVAlarm	Set
10/08/2012 09:06:00.114	Sim1_Alarm	Sim1 Alarm	System	Info	CVAlarm	Cleared
10/08/2012 09:05:19.258	Sim1_Alarm	Sim1 Alarm	Default_User A...	Info	CVAlarm	Ack
10/0		Sim1 Alarm	System	Warning	CVAlarm	Set
10/0		Sim1 Alarm	System	Warning	CVAlarm	Set
10/0		Sim1 Alarm	System	Info	CVAlarm	Cleared
10/05/2012 17:55:11.242	Sim1_Alarm	Sim1 Alarm	System	Warning	CVAlarm	Set
10/05/2012 17:55:00.087	Sim1_Alarm	Sim1 Alarm	System	Info	CVAlarm	Cleared
10/05/2012 17:54:11.133	Sim1_Alarm	Sim1 Alarm	System	Warning	CVAlarm	Set
10/05/2012 17:54:00.263	Sim1_Alarm	Sim1 Alarm	System	Info	CVAlarm	Cleared
10/05/2012 17:53:11.307	Sim1_Alarm	Sim1 Alarm	System	Warning	CVAlarm	Set
10/05/2012 17:53:00.152	Sim1_Alarm	Sim1 Alarm	System	Info	CVAlarm	Cleared
10/05/2012 17:52:11.480	Sim1_Alarm	Sim1 Alarm	System	Warning	CVAlarm	Set
10/05/2012 17:52:00.325	Sim1_Alarm	Sim1 Alarm	System	Info	CVAlarm	Cleared
10/05/2012 17:51:11.653	Sim1_Alarm	Sim1 Alarm	System	Warning	CVAlarm	Set
10/05/2012 17:51:00.500	Sim1_Alarm	Sim1 Alarm	System	Info	CVAlarm	Cleared
10/05/2012 17:50:11.096	Sim1_Alarm	Sim1 Alarm	System	Warning	CVAlarm	Set
10/05/2012 17:50:00.673	Sim1_Alarm	Sim1 Alarm	System	Info	CVAlarm	Cleared
10/05/2012 17:49:11.270	Sim1_Alarm	Sim1 Alarm	System	Warning	CVAlarm	Set
10/05/2012 17:49:00.116	Sim1_Alarm	Sim1 Alarm	System	Info	CVAlarm	Cleared

Show Normal
Show Event Sequence

Figure #16.8

TimeStamp	Alarm Name	Alarm Description	Source	Severity	Groups	Status
00/00/0000 00:00:00.000	Sim1_Alarm	Sim1 Alarm	System	Warning	CVAlarm	Set
00/00/0000 00:00:48.672	Sim1_Alarm	Sim1 Alarm	System	Info	CVAlarm	Cleared
00/00/0000 00:00:59.827	Sim1_Alarm	Sim1 Alarm	System	Warning	CVAlarm	Set
00/00/0000 00:01:48.783	Sim1_Alarm	Sim1 Alarm	System	Info	CVAlarm	Cleared
00/00/0000 00:01:59.653	Sim1_Alarm	Sim1 Alarm	System	Warning	CVAlarm	Set
00/00/0000 00:02:48.607	Sim1_Alarm	Sim1 Alarm	System	Info	CVAlarm	Cleared
00/00/0000 00:02:59.762	Sim1_Alarm	Sim1 Alarm	System	Warning	CVAlarm	Set
00/00/0000 00:03:49.165	Sim1_Alarm	Sim1 Alarm	System	Info	CVAlarm	Cleared
00/00/0000 00:03:59.601	Sim1_Alarm	Sim1 Alarm	System	Warning	CVAlarm	Set
00/02/0000 15:12:59.789	Sim1_Alarm	Sim1 Alarm	System	Warning	CVAlarm	Set
00/02/0000 15:13:07.778	Sim1_Alarm	Sim1 Alarm	Default_User A...	Info	CVAlarm	Ack
00/02/0000 15:13:48.634	Sim1_Alarm	Sim1 Alarm	System	Info	CVAlarm	Cleared
00/02/0000 15:13:59.616	Sim1_Alarm	Sim1 Alarm	System	Warning	CVAlarm	Set

Figure #16.9

Implementation

Hourly data range assumed. The single tag is considered.

Selecting Time Points

The interval will be from T1 to T2.

For T1 the following 2 cases:

T1 equals exact hours

T1 is between exact hours

And it is similar for T2. The points to be included into the output for those cases are as in Example 1 and 2 below:

Example 1

T1 = 10:00:00.000

T2 = 15:00:00:000

The output will be as follows:

Timestamp	Name	Max/Min/Sec
11:00:00.000	xxx	xxx
12:00:00.000	xxx	xxx
13:00:00.000	xxx	xxx
14:00:00.000	Xxx	xxx
15:00:00.000	Xxx	xxx

Example 2

T1 = 10:00:00.100

T2 = 15:01:00:000

The output will be as shown in the table below:

Timestamp	Name	Max/Min/etc.
12:00:00.000	Xxx	xxx
13:00:00.000	Xxx	xxx
14:00:00.000	Xxx	xxx
15:00:00.000	Xxx	xxx

In the second example, the data for 11:00 will not be shown because some data for the time range from 10:00 to 11:00 is missing (just for 100 milliseconds in this case, but nevertheless).

Calculating Aggregates

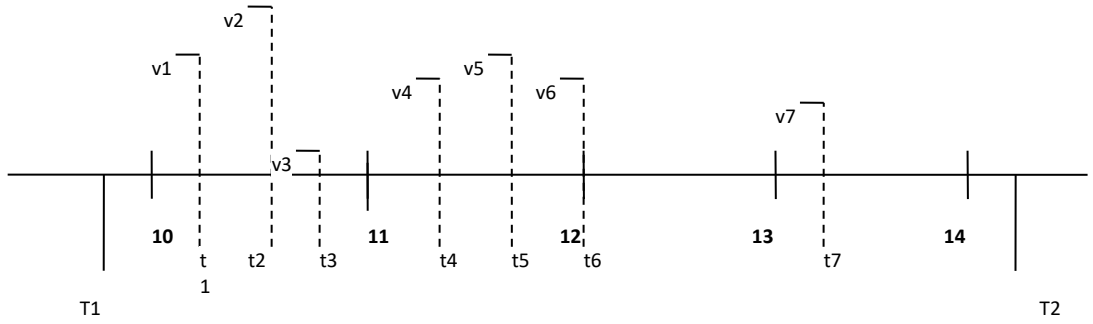


Figure #16.10

So, we have time range from sometime before 10:00 and sometime after 14:00. The History table contains records for times t1, . t7, and their corresponding values v1, … v7.

The table below shows how Max, Min SimpleAVG and WightedAVG are calculated.

Timestamp	Max	Min	SimpleAVG	WeightedAVG
11	Max(v1,v2,v3)	Min (v1, v2, v3)	(v1+v2+v3) / 3	(v1*(t2-t1) + v2*(t3-t2) + v3*(11-t3)) / (11 – t1)
12	Max (v4, v5)	Min (v4, v5)	(v4+v5)/2	(v3*(t4-11) + v4*(t5-t4)+v5(12-v5)) / (12 - 11)
13	v6 (this point belongs to interval from 12 to 13)	v6 (this point belongs to interval from 12 to 13)	v6	v6 * (13-t6) / (13 – 12)
14	v7	v7	v7	(v7 * (t6 - 13) + v7 * (14 – t7)) / (14 – 13)

Drawbacks

Calculation of the Min, Max, and SimpleAvg does not have a clear mathematical meaning. Just formal values over sample points are calculated.

Let us consider Min value of the tag over the interval from 11 to 12. The above formula proceeds from the assumption that on that time interval the tag had just values v4 and v5 (v6 belongs to the next interval). The value there must be somewhere between v3 and v4; hence, the minimal tag value is less than v4 as the current example shows.

So, when telling about mathematically-justified approach, one should consider statistical assessment of tag value changes based on the sample points provided in the History table. Sure, for this one must first interpolate function describing tag value change over the time.

Timestamp	Name	Max	Min	SimpleAVG	WeighteAVG
11	For 11 there is no data because we do not have sufficient information for the interval from 10 to 11. WeigtedAVG requires no correction.				
12		Max (v3, v4, v5)	Min (v3, v4, v5)	(v3+v4+v5)/3	(v3*(t4-11) + v4*(t5-t4)+v5(12-v5)) / (12 - 11)
13		v6 (this point belongs to interval from 12 to 13)	v6 (this point belongs to interval from 12 to 13)	v6	v6 * (13-t6) / (13 – 12)
14		max (v6, v7)	min (v6, v7)	(v6 + v7) / 2	(v7 * (t6 - 13) + v7 * (14 – t7)) / (14 – 13)

SECTION 17

CVPLOT ActiveX Object

Follow these steps to configure the CVPLOT ActiveX Object:

1. Select **Plot** Object from **Object Library** | **Data Controls** folder. See A in Figure #17.1).
2. Draw the object on ClearView SCADA screen.

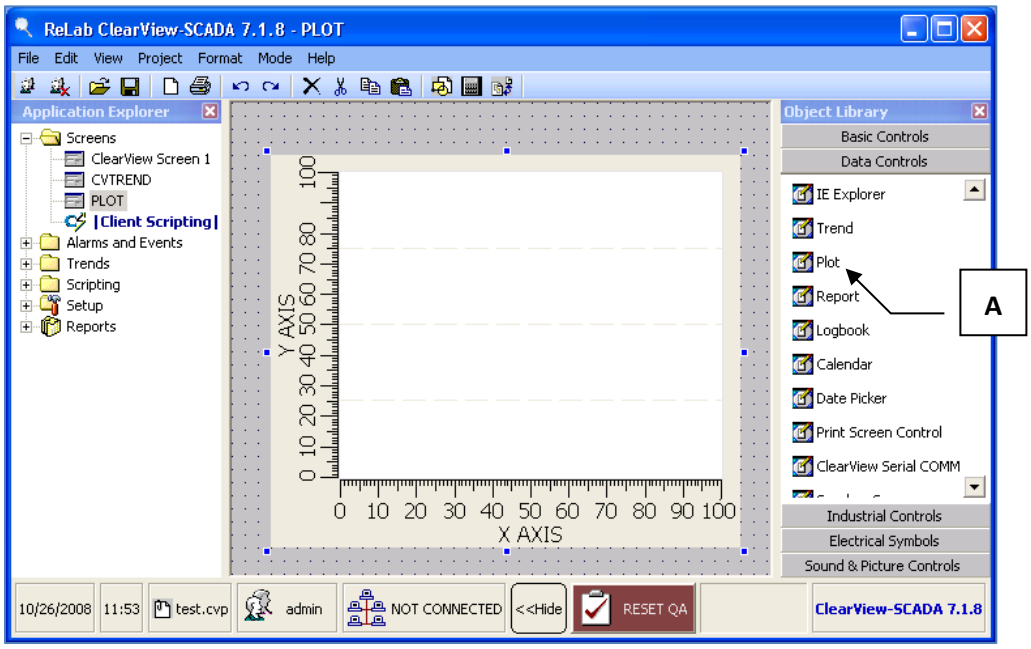


Figure #17.1

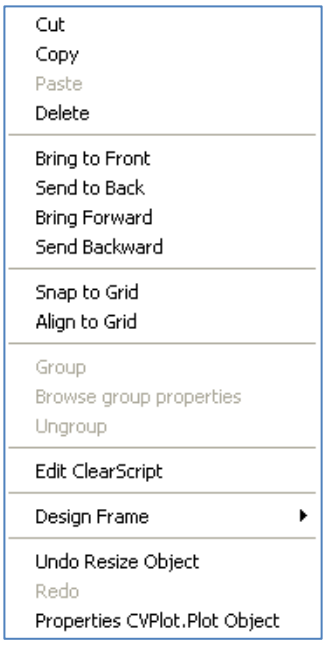


Figure #17.2

3. Right-click CVPLOT object and select **Properties CVPlot.Plot Object.**

CVPLOT Color Properties

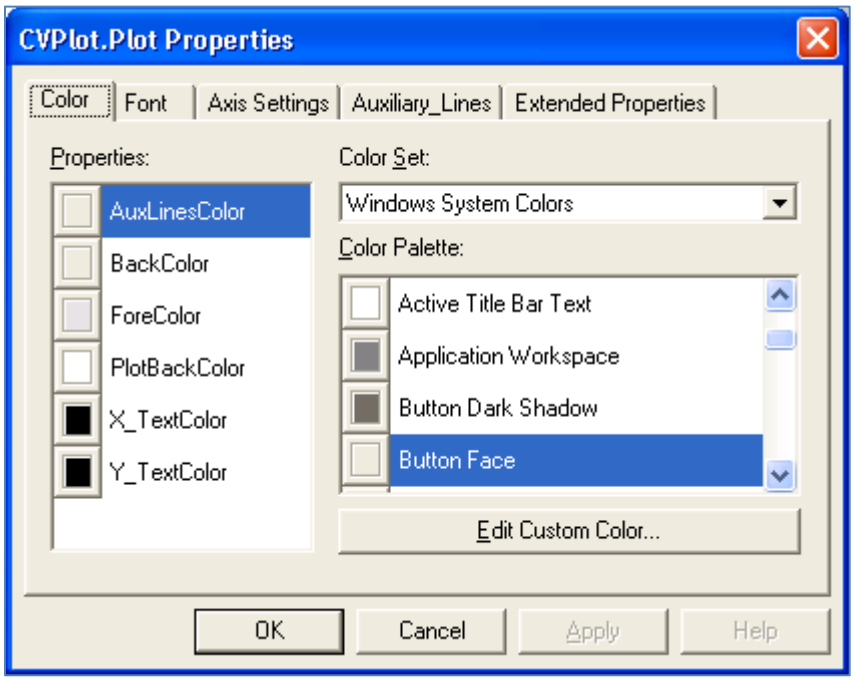


Figure #17.3

Property	Property Description	Type	Code
AuxLinesColor	Set's or returns auxiliary line color	Long (OLE Color)	Plot1.AuxLinesColor = 255
BackColor	Set's or returns control back color line color	Long (OLE Color)	Plot1.BackColor = 255
ForeColor	Set's or returns the plot frame color	Long (OLE Color)	Plot1.ForeColor = 255
PlotBackColor	Set's or returns the plot back color	Long (OLE Color)	Plot1.BackColor = 255
X_TextColor	Set's or returns the color of the X-Axis text color	Long (OLE Color)	Plot1.X_TextColor = 255
Y_TextColor	Set's or returns the color of the Y-Axis text color	Long (OLE Color)	Plot1.Y_TextColor = 255

CVPLOT User Scale

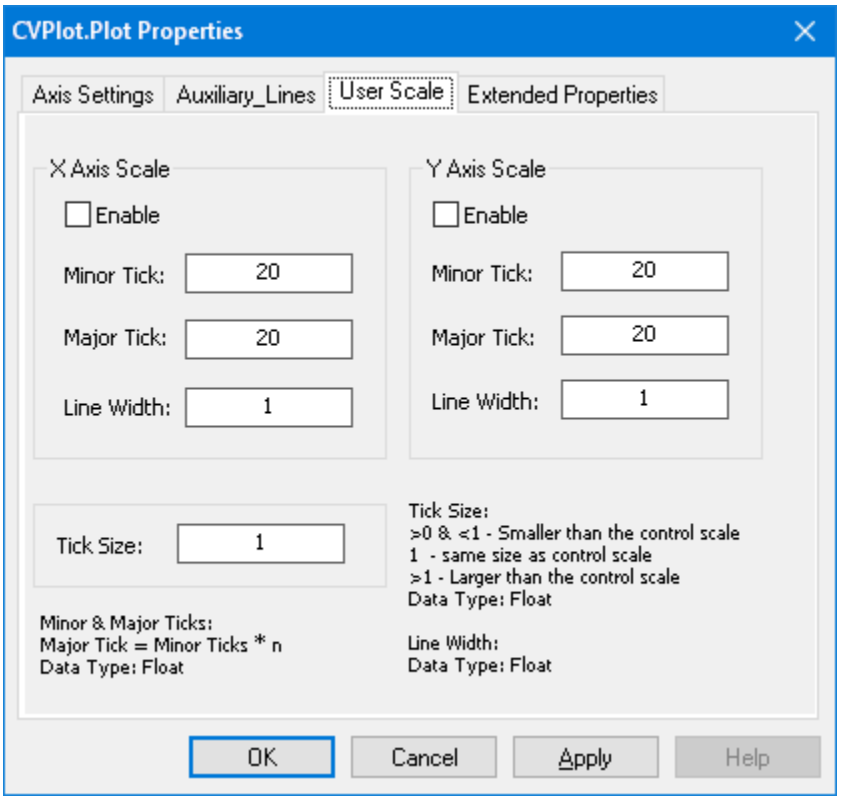


Figure #17.4

Property	Property Description	Type
X-Axis Scale Enable	Enables X-Axis Scale	Boolean
X-Axis Minor Tick	Specifies X-Axis Minor Tick value	Integer
X-Axis Major Tick	Specifies X-Axis Major Tick value	Integer
X-Axis Line Width	Specifies X-Axis Line Width value	Float
Y-Axis Scale Enable	Enables Y-Axis Scale	Boolean
Y-Axis Minor Tick	Specifies Y-Axis Minor Tick value	Integer
Y-Axis Major Tick	Specifies Y-Axis Major Tick value	Integer
Y-Axis Line Width	Specifies Y-Axis Line Width value	Float
Y-Axis Tick Size	Specifies Tick Size value	Float

CVPLOT Axis Settings Properties

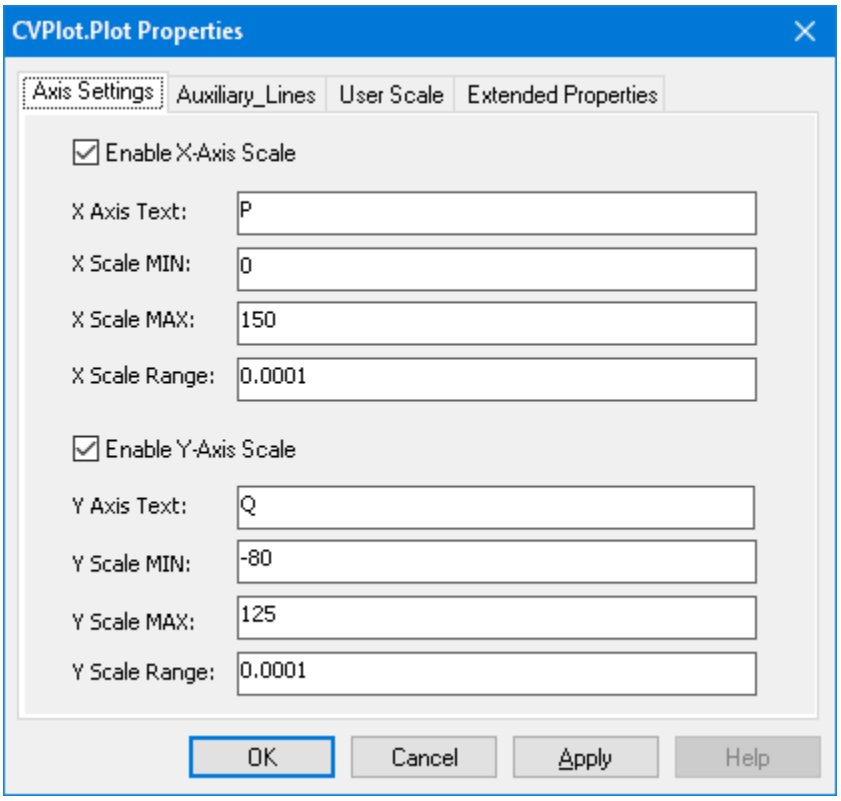


Figure #17.5

Property	Property Description	Type	Code
X_ScaleText	Set's or returns X Axis text	String	Plot1.X_ScaleText = "X-Axis"
X_MinScale	Set's or returns X Axis minimum scale	Double	Plot1.X_MinScale = 0
X_MaxScale	Set's or returns X Axis maximum scale	Double	Plot1.X_MaxScale = 100
XScaleRange	Set's or returns X Axis scale range	Double	Plot1.XScaleRange = 10
Y_ScaleText	Set's or returns Y Axis text	String	Plot1.Y_ScaleText = "X-Axis"
Y_MinScale	Set's or returns Y Axis minimum scale	Double	Plot1.Y_MinScale = 0
Y_MaxScale	Set's or returns Y Axis maximum scale	Double	Plot1.Y_MaxScale = 100
YScaleRange	Set's or returns Y Axis scale range	Double	Plot1.YScaleRange = 10

CVPLOT Auxiliary Lines Properties

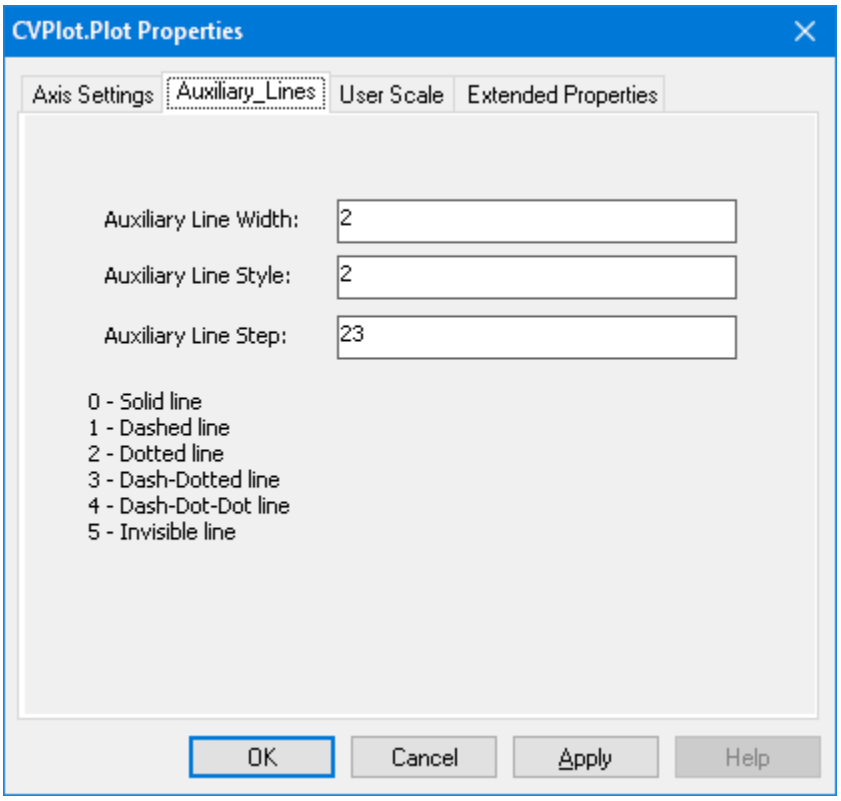


Figure #17.6

Property	Property Description	Type	Code
AuxLinesWidth	Set's or returns auxiliary line width	Integer	Plot1.AuxLinesWidth = 0
AuxLinesStyle	Set's or returns auxiliary line style	Integer	Plot1.AuxLinesStyle = 1 0 - Solid line 1 - Dashed line 2 - Dotted line 3 - Dash-Dotted line 4 - Dash-Dot-Dot line 5 - Invisible line

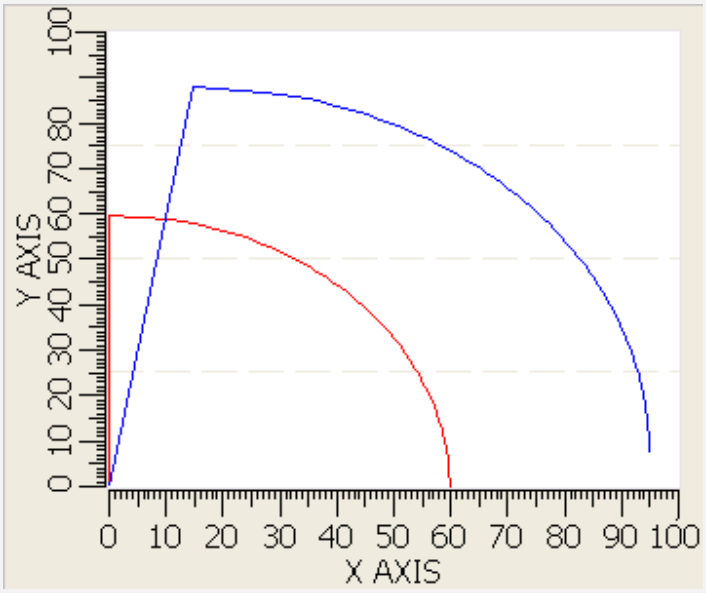
CVPLOT SetCurve Method

SetCurve – draws the user-defined curve:

SetCurve (Pen_Num As Integer, Pen_Color As Long, Pen_LineType As Integer, ByRef X, ByRef Y, Optional LineWidth As Integer)

Parameter	Parameter Description	Type
Pen_Num	Specifies the curve number (0 to 5)	Integer
Pen_Color	Specifies the curve color	Long (OLE Color)
Pen_LineType	Specifies the curve line type 0 - Solid line 1 - Dashed line 2 - Dotted line 3 - Dash-Dotted line 4 - Dash-Dot-Dot line 5 - Invisible line	Integer
X	Specifies array of curve X coordinates	Array of Double values
Y	Specifies array of curve Y coordinates	Array of Double values
LineWidth	Specifies curve width (1-16 points)	Integer (Optional)

```
Dim i
Dim n(100)
Dim m (100)
R = CDbl (60)
For i = 0 To 360
   n(i) = Sqr (R ^ 2 - i ^ 2)
   m(i) = Sqr (R ^ 2 - (Sqr (R ^ 2 - i ^ 2) ^ 2))
Next
Plot1.SetCurve 0, vbRed, 0, n, m
R = CDbl (80)
For i = 0 To 360
   n(i) = Sqr (R ^ 2 - i ^ 2) + 15
   m(i) = Sqr (R ^ 2 - (Sqr (R ^ 2 - i ^ 2) ^ 2)) + 8
Next
Plot1.SetCurve 1, vbBlue, 0, n, m, 2
```



CVPLOT SetPoint Method

SetPoint – draws the user-defined point:

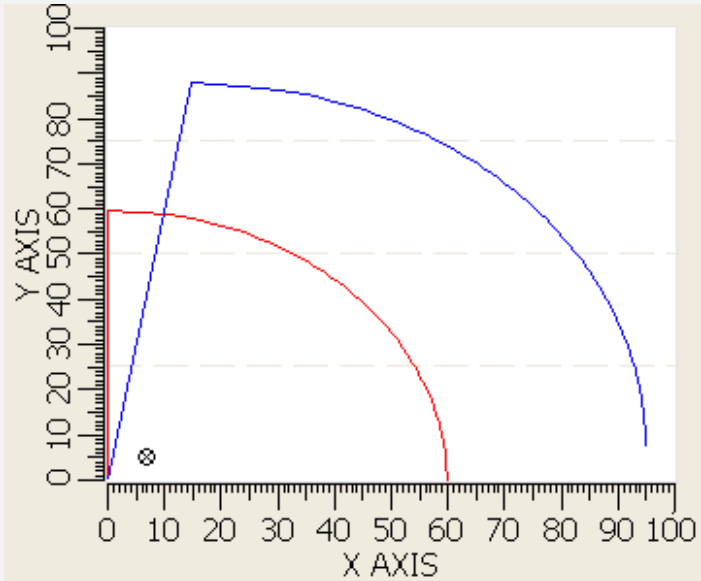
SetPoint (Point_Num As Integer, Point_Color As Long, Point_Type As Integer, Point_Size As Double, ByRef X, ByRef Y)

Parameter	Parameter Description	Type
Point_Num	Specifies the point (symbol) number (6 to 9)	Integer
Pont_Color	Specifies the point (symbol) color	Long (OLE Color)
Point_Type	Specifies the point (symbol) type -1 no symbol 0 △ 1 □ 2 ◇ 3 ○ 4 + 5 × 6 ✳ 7 ⊞ 8 ⊠ 9 ⌖ 10 ⊕ 11 ⊗	Integer
Point_Size	Specifies the size of the point	Float
X	Specifies point X coordinates	Double
Y	Specifies point Y coordinates	Double

x = Abs(Rnd(50)) * 10

y = Abs(Rnd(50)) * 10

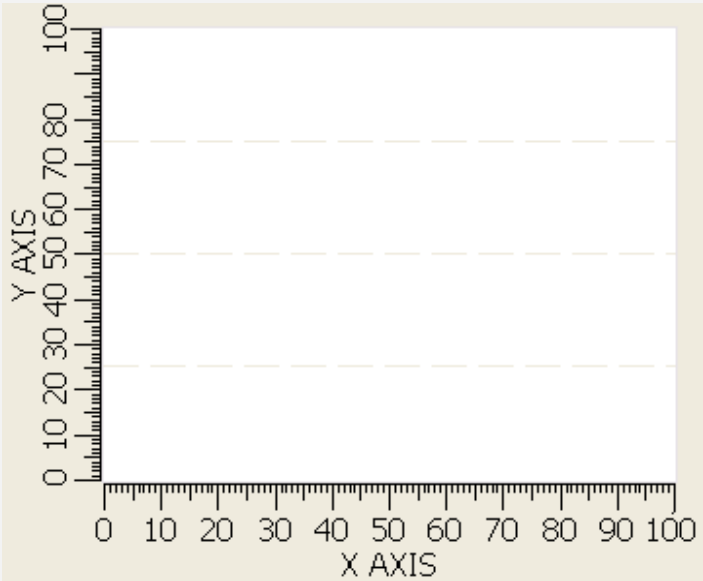
Plot1.SetPoint 6, vbGreen, 11, 0.7, x, y



CVPLOT ClearData Method

ClearData – clears the user defined curve/point:

ClearData (Point_Num As Integer)

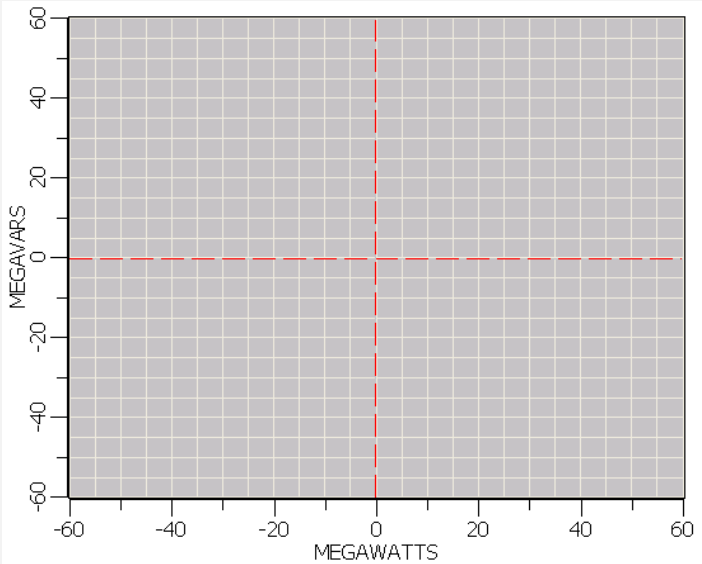
Parameter	Parameter Description	Type
Point_Num	Specifies the curve/point number (0 to 9)	Integer
Plot1.ClearData 0 Plot1.ClearData 1 Plot1.ClearData 3 Plot1.ClearData 4 Plot1.ClearData 5 Plot1.ClearData 6		

CVPLOT UserScaleXY Method

UserScaleXY – draws X and Y coordinates based on user inputs:

UserScaleXY (XY, Pen_Color, Pen_LineType, LineWidth)

Parameter	Parameter Description	Type
XY	Specifies the X and Y coordinates XY = 0 X Only XY = 1 Y Only XY = 2 X and Y	Integer
Pen_Color	Specifies the color	Long (OLE Color)
Pen_LineType	Specifies the line type 0 - Solid line 1 - Dashed line	Integer

	2 - Dotted line 3 - Dash-Dotted line 4 - Dash-Dot-Dot line 5 - Invisible line	
LineWidth	Specifies the line width	Integer (Optional)
Plot1.UserScaleXY 2, vbRed, 1, 1		

CVPLOT RemoveUserScaleXY Method

UserScaleXY – removes X and Y coordinates:

RemoveUserScaleXY(XY)

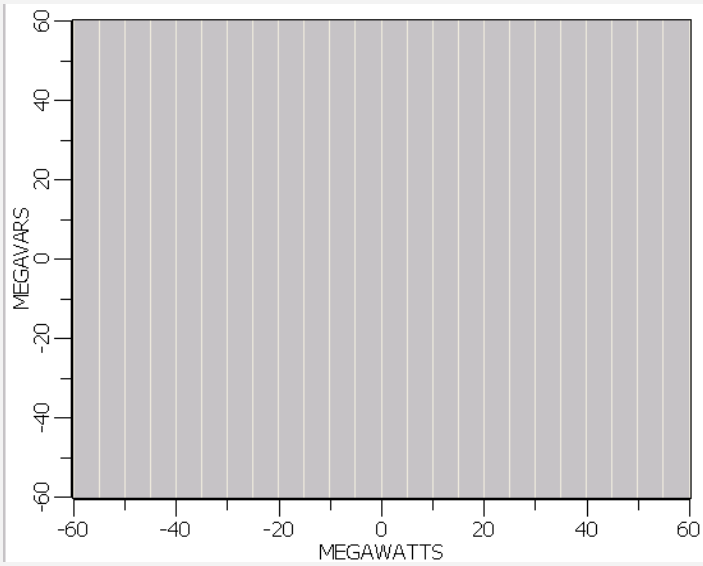
Parameter	Parameter Description	Type
XY	Specifies the X and Y coordinates XY = 0 X Only XY = 1 Y Only XY = 2 X and Y	Integer

CVPLOT YGridLines Method

YGridLines – Draws (Y-Axis) user definite auxiliary lines:

YGridLines (AuxLinesEn, AuxLinesStart, AuxLinesStep, AuxLinesColor, AuxLinesStyle)

Parameter	Parameter Description	Type

AuxLinesEn	Enables drawing of auxiliary lines AuxLinesEn = True (draw lines) AuxLinesEn = False (remove lines)	Boolean
AuxLinesStart	Y Axis auxiliary line start point	Double (Optional)
AuxLinesStep	Y Axis auxiliary line step value	Double (Optional)
AuxLinesColor	Specifies the line color	Long (Optional)
AuxLinesStyle	Specifies the line type 0 - Solid line 1 - Dashed line 2 - Dotted line 3 - Dash-Dotted line 4 - Dash-Dot-Dot line 5 - Invisible line	Integer
Plot1.YGridLines True, 0, 5, &H8000000F, 0		

CVPLOT XGridLines Method

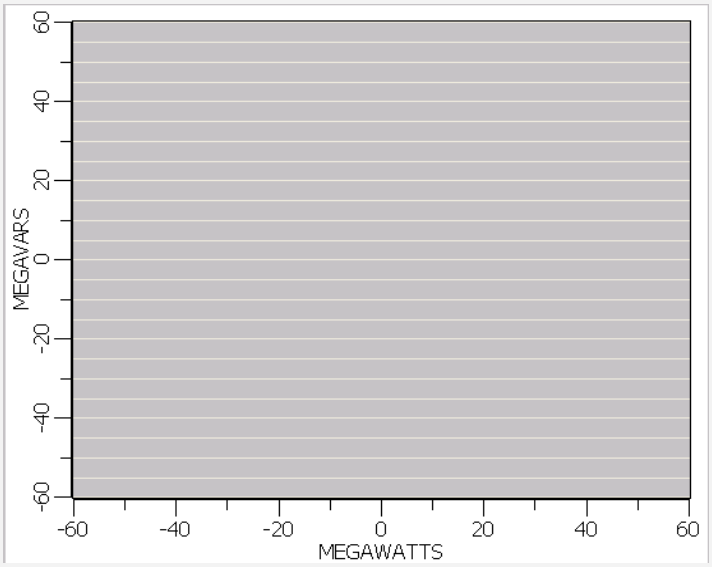
XGridLines – Draws (X-Axis) user definite auxiliary lines:

XGridLines (AuxLinesEn, AuxLinesStart, AuxLinesStep, AuxLinesColor, AuxLinesStyle)

Parameter	Parameter Description	Type

AuxLinesEn	Enables drawing of auxiliary lines AuxLinesEn = True (draw lines) AuxLinesEn = False (remove lines)	Boolean
AuxLinesStart	X Axis auxiliary line start point	Double (Optional)
AuxLinesStep	X Axis auxiliary line step value	Double (Optional)
AuxLinesColor	Specifies the line color	Long (Optional)
AuxLinesStyle	Specifies the line type 0 - Solid line 1 - Dashed line 2 - Dotted line 3 - Dash-Dotted line 4 - Dash-Dot-Dot line 5 - Invisible line	Integer (Optional)

Plot1.XGridLines True, 0, 5, &H8000000F, 0



SECTION 18

CVTREND ActiveX Object

Follow these steps to configure the CVTREND ActiveX Object:

1. Select **Trend** Object from **Object Library** | **Data Controls** folder. (See **A** in Figure #18.1)
2. Draw the object on ClearView SCADA screen.

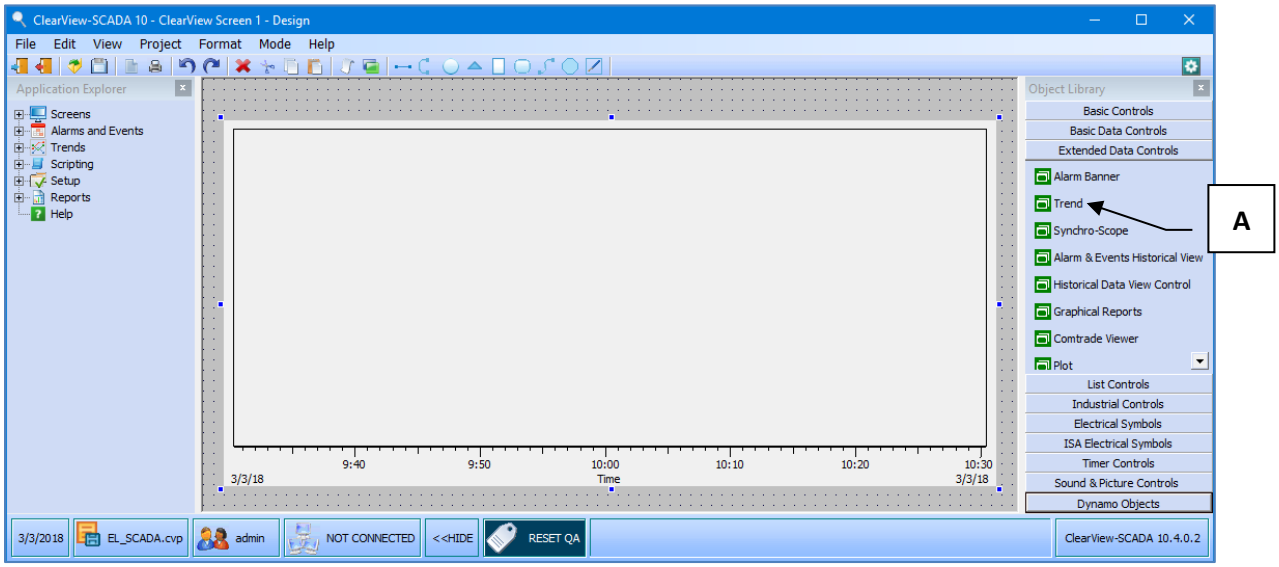


Figure # 18.1

3. Right click **CVTREND** object and select the **CVTrend1.CVTrend Object** property.

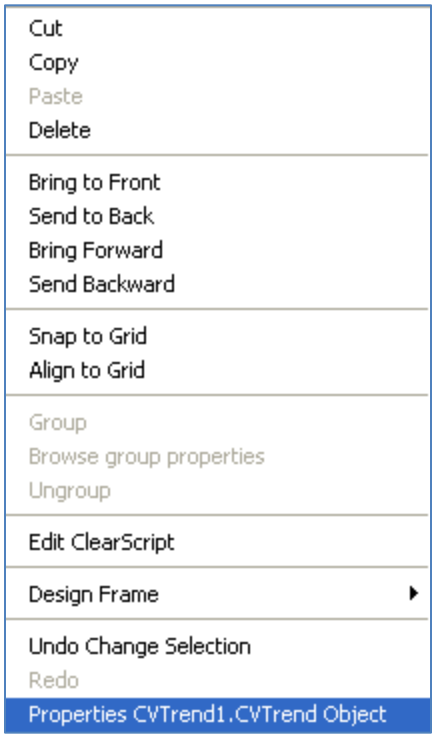


Figure #18.2

CVTREND Default Settings Properties

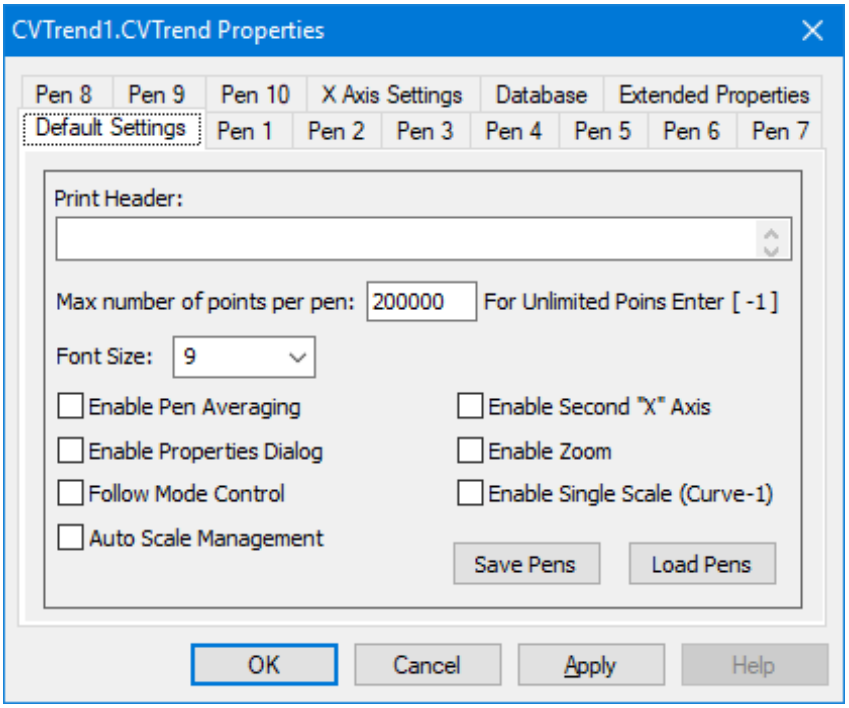


Figure #18.3

Property	Property Description	Type	Code
Print Header	Specifies the print header (title).	String	CVTrend1.PrintTitle = "Enter the title here"
Maximum number of points per pen	When the lengthy measurement is monitored, it is necessary to limit the number of points per pen. If the number of the data points is set to unlimited (-1), then the application itself must control the memory to avoid an overflow.	Long	CVTrend1.MaxPoints = -1
Enable Pen Averaging	When a number of approximately 3840 data points has been reached or exceeded, the details of a curve are no longer visible because it is restricted by resolution of the output device. To obtain useful graphs, the control supports averaging routine.	Boolean	CVTrend1.EnableCurveAVG = True (or False)
Enable Property Dialog	Enables or disables property dialog box in run-time.	Boolean	CVTrend1.EnablePropertiesDlg= True (or False)
Enable Follow Mode Control	Enables or disables property “X” axis follow mode (button).	Boolean	CVTrend1.EnableFollowModeBT = True (or False)
Enable Second “X” Axis	Enables or disables second “X” axis.	Boolean	CVTrend1.EnableFollowModeBT = True (or False)
Enable Zoom	Enables or disables area zooming.	Boolean	CVTrend1.EnableZoom = True (or False)
Enable Single Scale	Enables or disables single “Y” axis scale. Enabling of the single scale will disable manual scale adjustment (the “Y” axis limits must be set in design mode). Note: The “Save Trend” & “Load Trend” described below (“Methods”)	Boolean	CVTrend1.EnableSingleScale= True (or False)

CVTREND Default Settings Methods

Property	Property Description	Type	Code
Save Pens	Saves current trend configuration	N/A	CVTrend1.SavePens ("C:\Test.trv")
Load Pens	Opens saved trend configuration	N/A	CVTrend1.LoadPens ("C:\Test.trv")

CVTREND Color Properties

Property	Property Description	Type	Code
Background_Color	Specifies the trend background color.	OLE_COLOR	CVTrend1.Background_Color = vbRed
X_Axis Color	Specifies the color of the X-Axis.	OLE_COLOR	CVTrend1. X_Axis_Color = 16576
Curve1_Color (1-10)	Specifies colors for each of the pens.	OLE_COLOR	CVTrend1. Curve1_Color = &H000040C0&

CVTREND Pen 1 to Pen 10 Properties

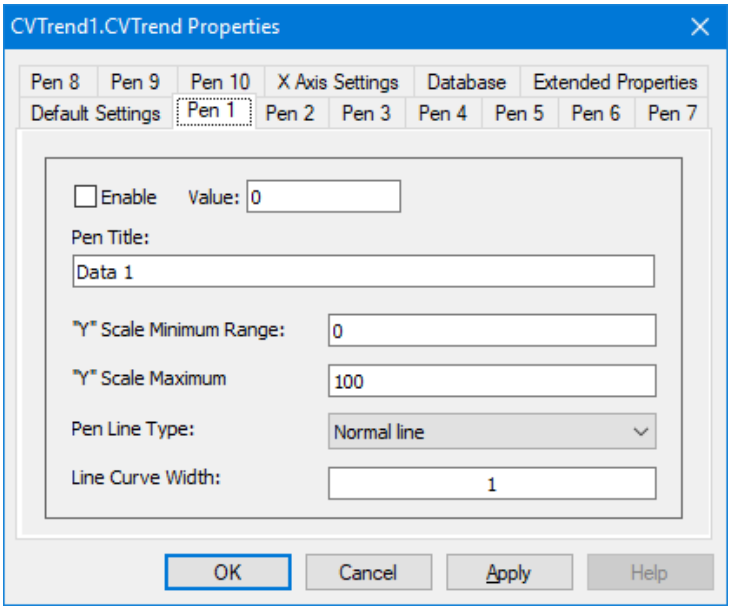


Figure #18.5

Property	Property Description	Type	Code
Enable	Enables a specified pen.	Boolean	CVTrend1.CurveEnable1 = True
Value	Specifies the current point value.	Double	CVTrend1.CurveValue1 = Tags.item("Sim1").Value
Print Header	Inputs data to print the trend header (title).	String	CVTrend1.PrintTitle = "Enter Title Here" ' *
"Y" Scale Minimum Range	Specifies minimum range of the "Y" axis.	Double	CVTrend1.YScaleRangeMin1 = 0

“Y” Scale Maximum Range	Specifies maximum range of the “Y” axis.	Double	CVTrend1.YScaleRangeMax1 = 100
Pen Line Type	Specifies pen line types	Integer	CVTrend1.Curve1_LineType = 1 ‘**
Line Curve Width	Specifies the thickness of the curve line (pen)	Integer	CVTrend1.LineCurveWidth1 = 1

Warning 1: The trend values can only be of a double data type. It is recommended you check the value before executing the script.

Warning 2: When the line width on the curve is not equal to 1px (default), the trend draw procedure is very complicated. This may lead to a considerable restriction of the system performence when large datasets are processed.

```
If IsNumeric(Tags.item("Sim1").Value) = True
  CVTrend1.CurveValue1 = Tags.item("Sim1").Value
End If
```

** Line Types

- Normal line = 0
- Dashed line = 1
- Dotted line = 2
- Dash-Dotted line = 3
- Dash-Dot-Dot line = 4
- Invisible line = 5

CVTREND X Axis Settings Properties

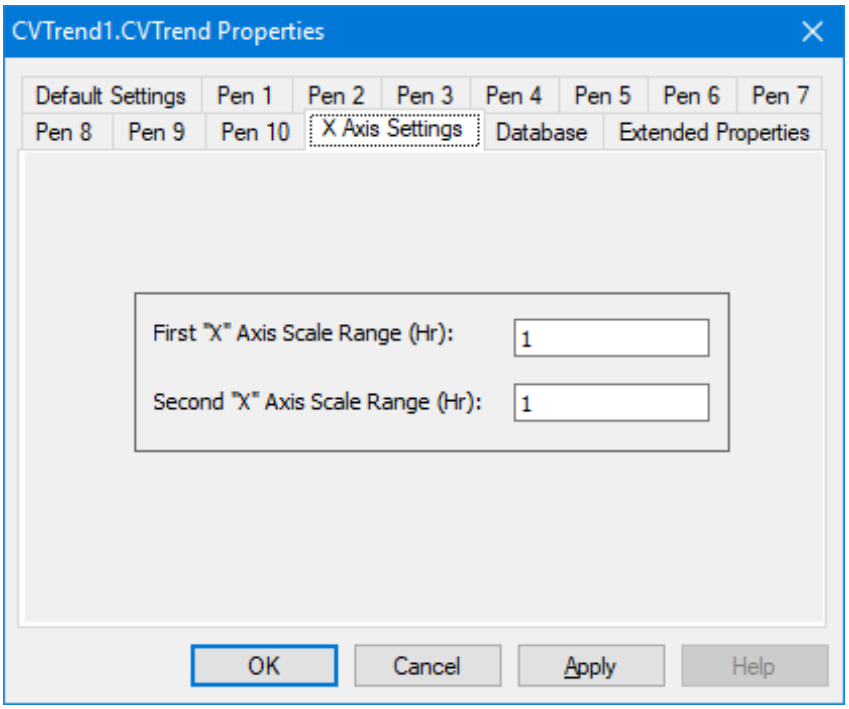


Figure # 18.6

Property	Property Description	Type	Code
1st "X" Axis Scale Range	Specifies 1st X-Axis scale range (Hr).	Double	CVTrend1.XScaleRange1 = 0.0166 '(one min)
2nd "X" Axis Scale Range	Specifies 2nd X-Axis scale range (Hr).	Double	CVTrend1.XScaleRange2 = 0.0166 '(one min)

CVTREND Database Properties

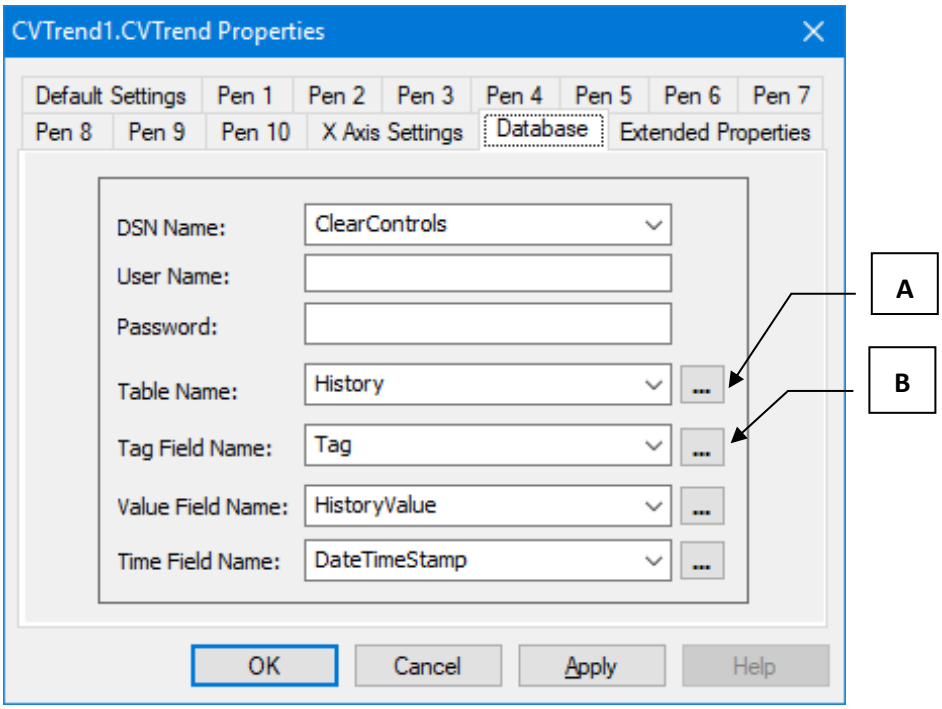


Figure #18.7

Note: Click **A** to refresh the table names. Click **B** to refresh table field names.

Property	Property Description	Type	Code
DSN Name	Specifies the ODBC data source name	String	CVTrend1.DSN_Name = "ClearControls"
User Name	Specifies the database user name	String	CVTrend1.User_Name = "Admin"
Password	Specifies the database password	String	CVTrend1.User_Password = "clearview"
Table Name	Specifies the table name	String	CVTrend1.Tag_Table = "History"
Tag Field Name	Specifies tag name filed	String	CVTrend1.Tag_Field = "Tag"
Value Field Name	Specifies value filed	String	CVTrend1.Value_Field = "HistoryValue"
Time Field Name	Specifies date/time filed	String	CVTrend1.Time_Field = "DateTimeStamp"

CVTREND Historical Trend Method

The historical trend method will access a previously configured database. It will also retrieve data based on specified tag (point) name.

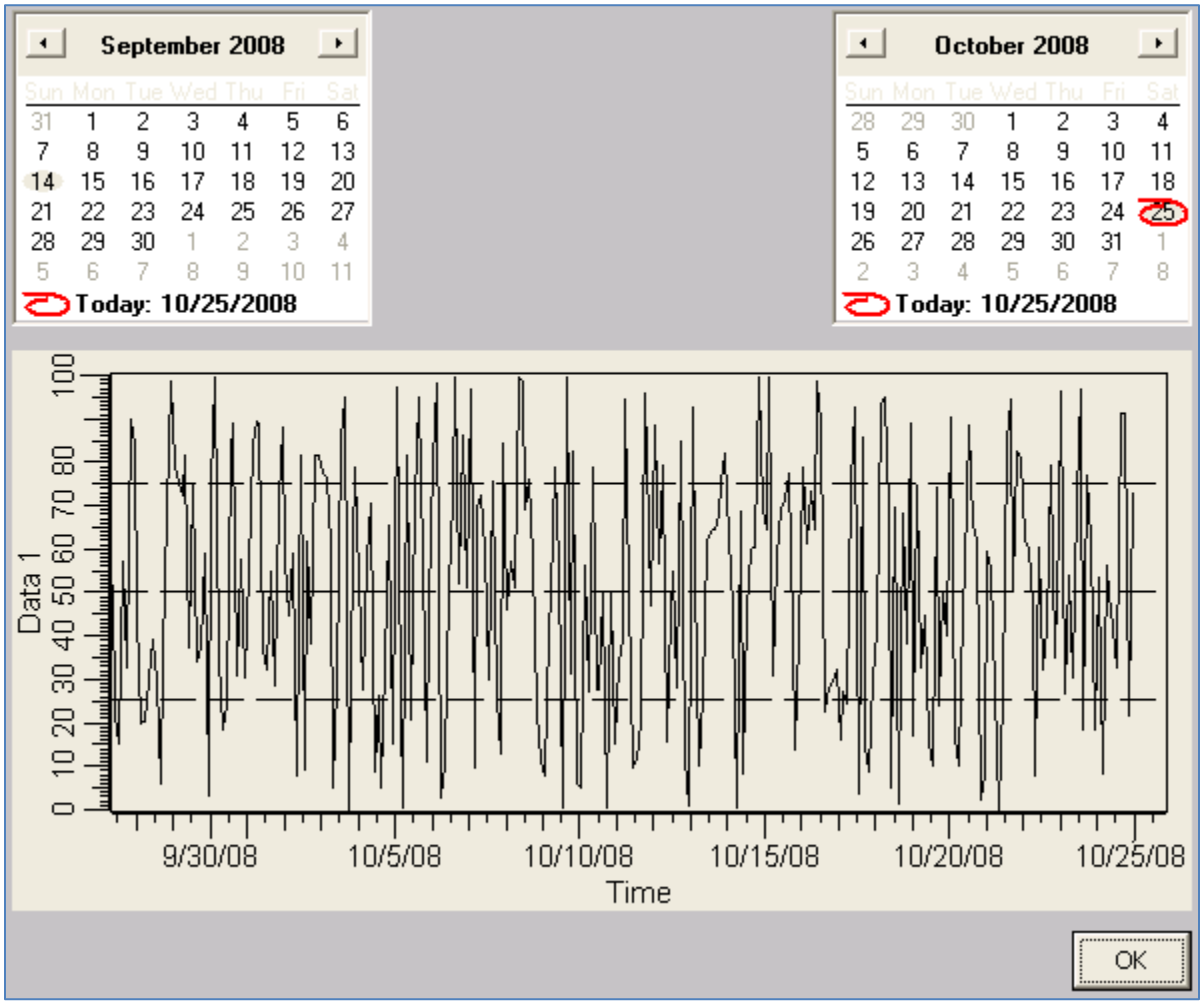


Figure #18.8

```
Private Sub ButtonCtrl1_MouseDown(Button, Shift, X, Y )
  CVTrend1.DBPoints 0, "Tag1", CDbl (ToDate.Value), CDbl (FromDate.Value), Trend_Error
End Sub
```

- **PenNumber** – specifies the pen (0 – 9) [Data Type: Integer]
- **Tag1** – Query tag name [Data Type: String]
- **ToDate.Value** – Max time-span for Tag1 query (double values only) [Data Type: Double]
- **FromDate.Value** – Min time-span for Tag1 query (double values only) [Data Type: Double]
- **CVTrend_Error** – error messages received from database connection [Data Type: String] – Optional

CVTREND Add Point Method

This method will add individual points to CVTREND control

CVTrend1.AddPoint 0, 10.5, 27.6, 39745.83460625, 39746.83460625

- **0** – specifies the pen (0 – 9) [Data Type: Integer].
- **10.5** – returns "Y" position of the point [Data Type: Double].
- **27.6** – returns "X" position of the point [Data Type: Double]
- **39745.83460625** – Sets the "X" scale maximum for X1 and X2 scales [Data Type: Double].
- **39746.83460625** – Sets the "X" scale minimum for X1 and X2 scales [Data Type: Double].

CVTREND Clear Trend Method

This method deletes all trend data points (trend clear).

CVTrend1.ClearTrend

CVTREND Delete Pen Method

This method deletes data for a specified pen (0 – 9).

CVTrend1.DeletePen (0)

CVTREND Print Trend Method

This method enables trend printing. Setting the PrintTrend to True will bring up the user's printer interface. Setting the PrintTrend to False will execute the print method to a default printer.

CVTrend1.PrintTrend (True)

CVTREND Lock Trend Updates Method

If the PrintTrend parameter is set to true the trend updates will be locked. Setting the lock updates to false will enable updates.

CVTrend1.TrendLockUpdate (True)

CVTREND Set Trend Auxiliary Lines Color

This method sets the colors of the auxiliary lines.
Parameter: Long Integer (OLE Color)

CVTrend1.SetTrendAuxLinesColor 255

CVTREND Set Ruler to OFF

This method hides the ruler.
Parameter: None

CVTrend1. Set_Ruler_OFF

CVTREND Set Data Window Background Color

This method Defines the background color of the data window as well as the display window for the ruler.
Parameter: Long Integer (OLE Color)

CVTrend1.Set_DW_Color 255

CVTREND Show Data Window

This method makes data window visible or invisible.
Parameter: False/0 – Sets Data Window to OFF, True/1– Sets Data Window to ON (visible)

CVTrend1. ShowDataWindow 1

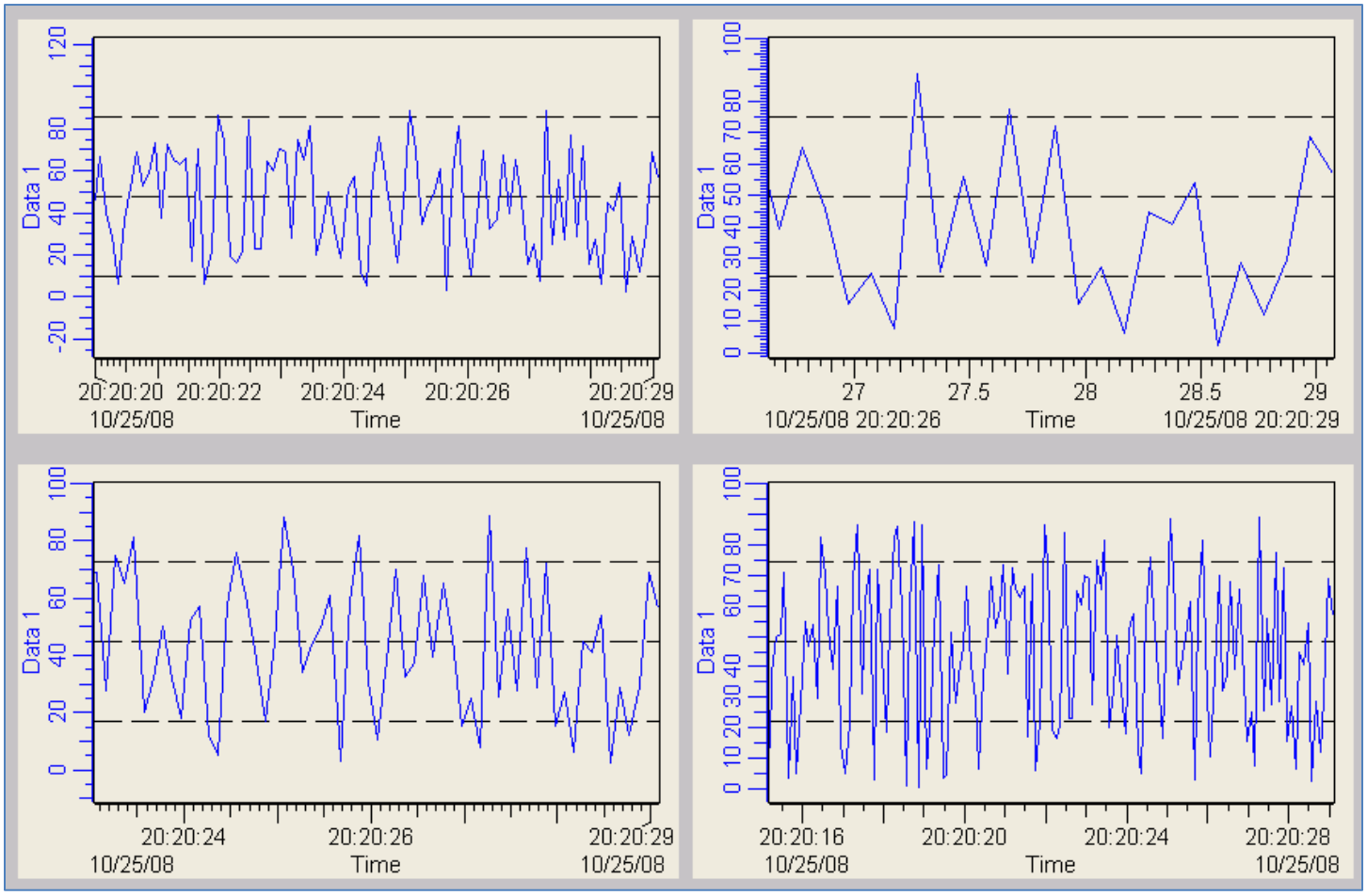


Figure #18.9

SECTION 19

Redundancy

Basic ClearView-SCADA Redundancy

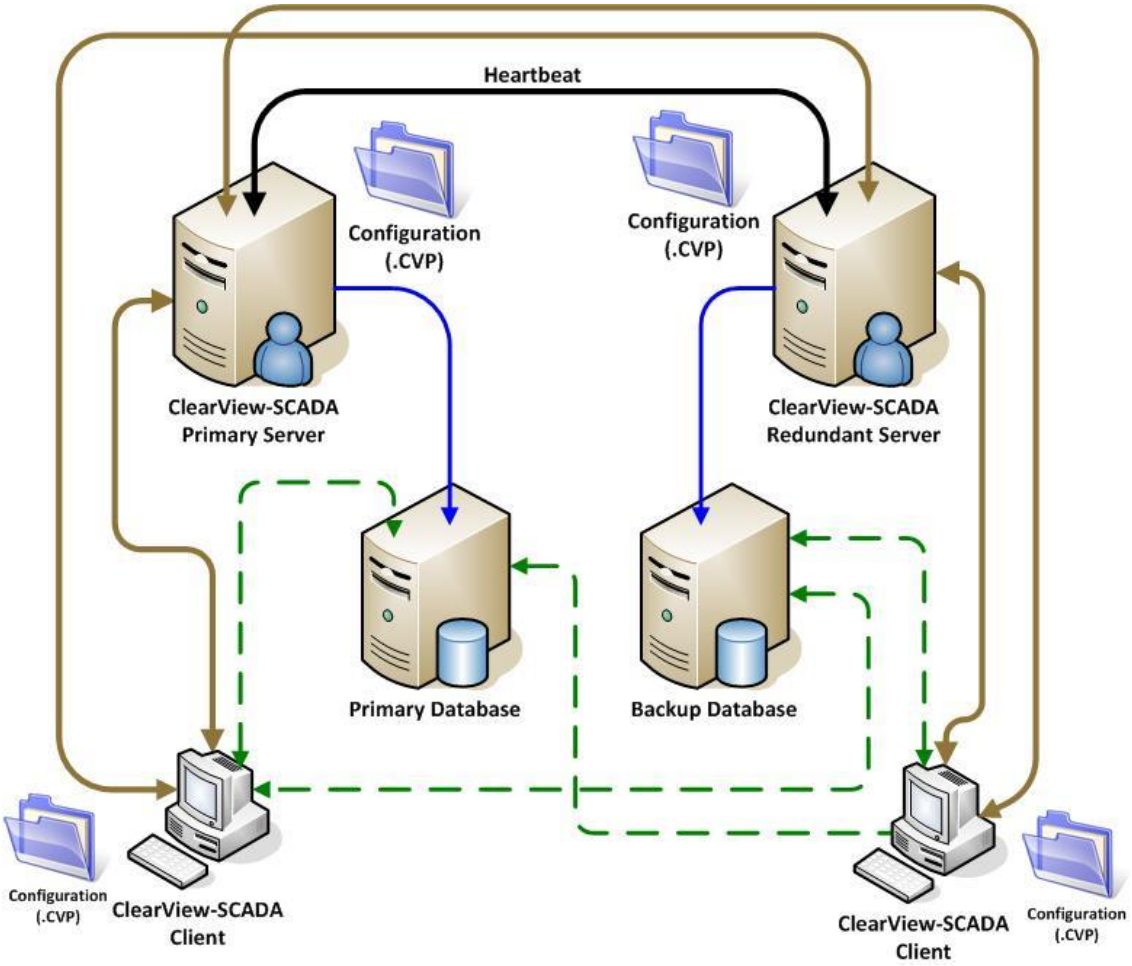


Figure #19.1

Shared Database ClearView-SCADA Redundancy

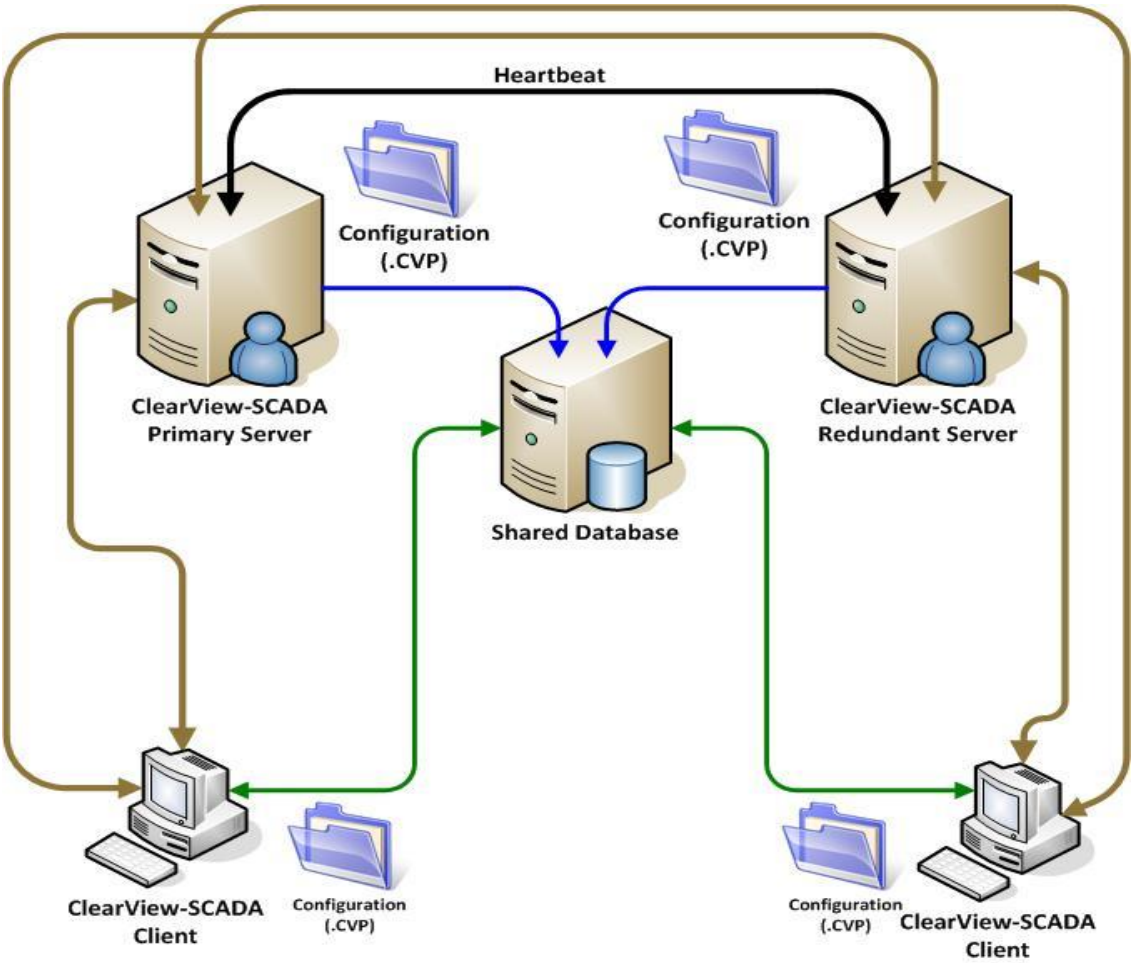


Figure #19.2

Description of Redundancy

Two ClearView-SCADA servers work in parallel; one of them, however, is configured to be the primary, while the other is configured to be the backup. At any given time only one server can be operating in active mode. The active server may be either of the servers. The decision as to which is the primary or backup as well as when to change is made via heartbeat exchange between servers. The following rules apply here:

ClearView-SCADA Server Redundancy

- A heartbeat connection is maintained by the primary server. It connects to the peer (backup server) upon startup and keeps trying to reconnect if the connection has been lost or cannot be established.
- Each server periodically sends heartbeats to the peer stating current mode of operation (active/inactive).
- If a server gets no heartbeat from a peer for some time, it considers the peer is unavailable and changes its own mode of operation to Active (if not currently Active).
- If a server is active, and its peer also reports its mode as active, the conflict is resolved in favor of the primary server. That is, the primary server remains active in this case, while the backup server becomes inactive. (See **Configuration Settings** on how to define and configure a primary server.)

- If the server is inactive, and the peer also reports its mode as inactive, the conflict is resolved in favor of the primary server. That is, the primary server steps up to active mode, while the backup server remains in inactive mode.

Database Redundancy

- On startup each ClearView-SCADA server would establish a connection to its corresponding database. If the database is not available the server would continue testing connection until connection is established. After the fourth attempt the server will show a window notifying that the connection is not established yet. The user will have an option to cancel connection attempts by pressing Ok button. If eventually the server connects to the database the notification window will be closed.
- After successful startup each ClearView-SCADA Servers periodically checks the health of its own database based on setting specified in project file “Database Connection Interval Timeout”. If connection to the database is lost and current status ClearView-SCADA server is Primary the state of the server would change to Backup and peer ClearView-SCADA server would be promoted to Primary.

Communication Link Redundancy (Tags)

- It is possible to configure field devices redundancy. The configuration file changes are required to “Bad Quality Count” & “Bad Quality Test”. “Bad Quality Count” determines the number of allowed bad tags. The “Bad Quality Test” determines how often ClearView-SCADA would test this condition.
- If after preset time currently primary ClearView-SCADA Server determines that the number of BAD Quality Tags exceeds by “Bad Quality Count” or more the number of bad quality tags on Backup ClearView-SCADA server it would demote itself to backup and currently backup server automatically would get a primary status.

Servers in active and inactive mode operate identically except that in the inactive mode:

- A server ignores tag write requests from ClearView-SCADA clients.
- A server ignores alarm acknowledgment requests from the ClearView-SCADA clients
- A server does not send tag updates to the ClearView-SCADA clients.
- A server does not send alarm changes to the ClearView-SCADA clients
- A server does not write history data to the database.

In order to make use of the basic redundancy, each ClearView-SCADA client keeps connections to both ClearView-SCADA servers. Since the inactive (backup) ClearView-SCADA server is in a kind of “idle mode,” the operation of the ClearView-SCADA client from the end-user perspective is just as if a single server were connected.

The difference between the General (primary and backup database) and Simple (shared or single database) configurations lies in the way the databases are used.

In General case, each server instance connects to its own database, while clients are connected to the database of the currently active server. When active server loses connection to its database and cannot reconnect for some preconfigured time, it becomes inactive, thereby triggering its peer to switch to the active mode. Then the client detects that the active server has changed and disconnects from the database of the previously active server; then the client connects to the database of the server that has become active.

In Simple case (the single database shared by servers), the deactivation of servers and reconnection of clients described above does not take place. Each component just tries to keep connected to the shared database.

Configuration Settings

Each client and server instance should have its own individual configuration, even though these configurations have much in common. The configurations should be consistent in order to make redundancy work correctly.

The following options from the *.cvp file are involved in configuring the basic redundancy and they must have values as described in the table below.

Name	Description	Used by Server	Used by Client
[ClientServer]			
ServerName	Host where Primary ClearView-SCADA server is running (see Note 1 below).	x	x
[ClientDB]			
DataProvider	One of Access, MSSQL, or Oracle depending upon the RDBMS type of the database used by the servers. Must be the same in all *.cvp files.	x	x
DataSource	ODBC datasource name configured for the ClearView-SCADA database on the host where corresponding component is running. The value may be different per component. For the server this is the datasource for the database used by this server. For the client this is the datasource for the database used by the Primary server.	x	x
OraUsername	User Login used by the components to connect to the above datasource.	x	x
OraPassword	User Password used by the components to connect to the above datasource.	x	x
DataSourceBack	Similar to the datasource except that for the server it is datasource of its peer for the client it is datasource of the Backup server **WARNING**: For the simple configuration datasource, primary and backup database MUST HAVE equal datasources		x
OraUsernameBack	User Login used by the components to connect to the above Backup datasource		x

Name	Description	Used by Server	Used by Client
OraPasswordBack	User Password used by the components to connect to the above Backup datasource		x
DBInterval	Defines how long (in milliseconds) ClearView-SCADA server should attempt connecting to its database before it changes its mode to Inactive.	x	
[ServerRedundancy]			
RedundancyEnabled	Must be 1 for all components	x	x
ServerName	Host where Backup ClearView-SCADA server is running (see Note 1 below).	x	x
[ServerMisc]			
HBInterval	Interval (in milliseconds) between heartbeats sent by server.	x	
HBTimeOut	Defines how long the server should wait for the subsequent heartbeat from the peer before stepping down to the inactive mode.	x	
BQTagsThreshold	Defines quantity of the BAD tags for redundancy switchover	x	
BQTagsInterval	Defines frequency by which ClearView-SCADA Server would check the difference in BAD tags between Primary & Redundant Server	x	

Note: Option values of [ClientServer]. ServerName and [ServerRedundancy] ServerName are used in the following way:

- **ClearView-SCADA Client** - for connecting to the primary and backup servers. In client's *.cvp file, the value may be a host that is pingable from the ClearView-SCADA client's host. User can specify an IP Address, Machine Name, or Domain Name of the host where corresponding ClearView-SCADA server is configured to run.

- **ClearView-SCADA Server** – to identify itself at runtime as primary or backup server by comparing itself against the computer name where it is running. Also, it is used by the primary server to establish a heartbeat connection with its peer. If IP address or domain name is used for this option value, the server attempts to resolve the computer name from that information. Depending, however, upon the configuration on the computer when the server is running, this may not always succeed. In particular, "localhost" and "127.0.0.1" should never be used. Hence, it is highly recommended you use short computer names for the discussed option values.

Configuring Redundancy using GUI

The identified setting from above may be configured on the Project Settings dialog box of the ClearView-SCADA client. Below are examples of configurations.

Simple (shared database) Configuration

The configuration may be the same for all components provided that hosting computers have corresponding datasource correctly configured.

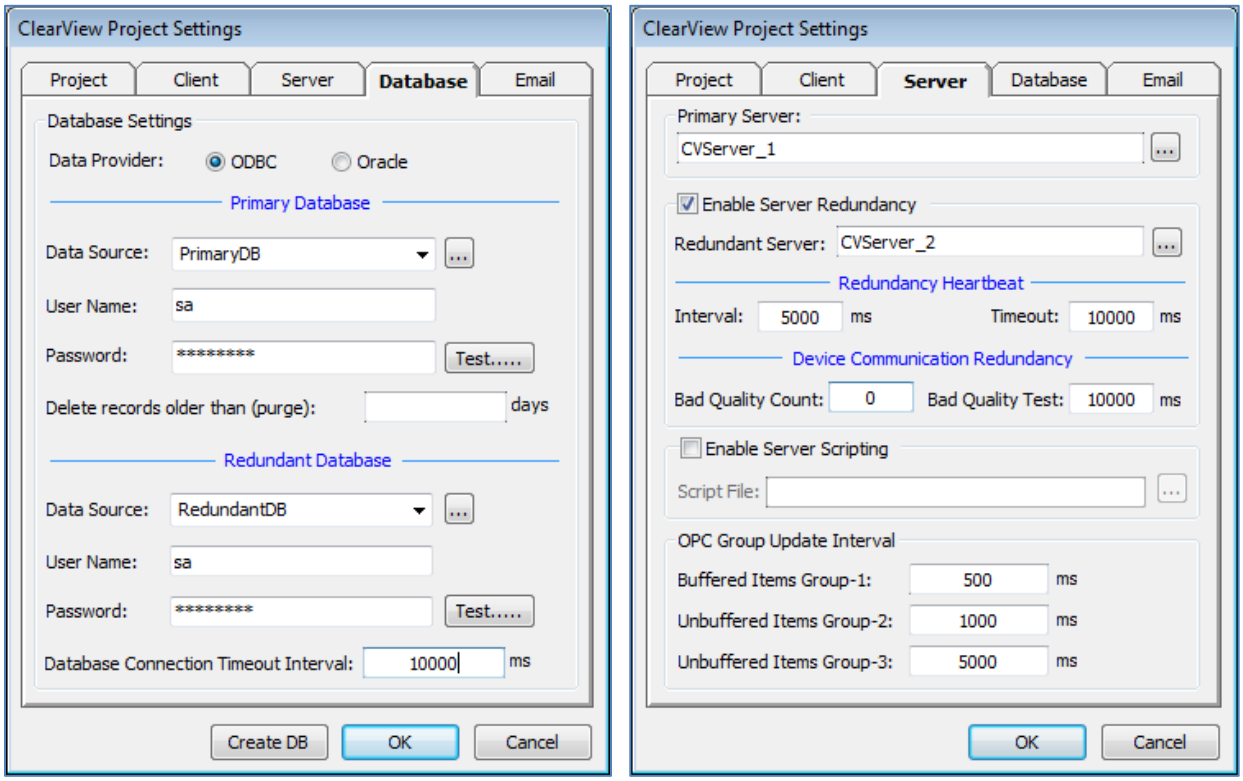


Figure #19.3

General (Primary with Backup Database) Configuration

The configuration process for the General configuration differs only by the settings for the database page. There are two configurations, one for the ClearView-SCADA Client and the primary ClearView-SCADA Server and another one for the

backup ClearView-SCADA Server. For the client, primary server and backup server the settings should be the same (Figure #19.4).

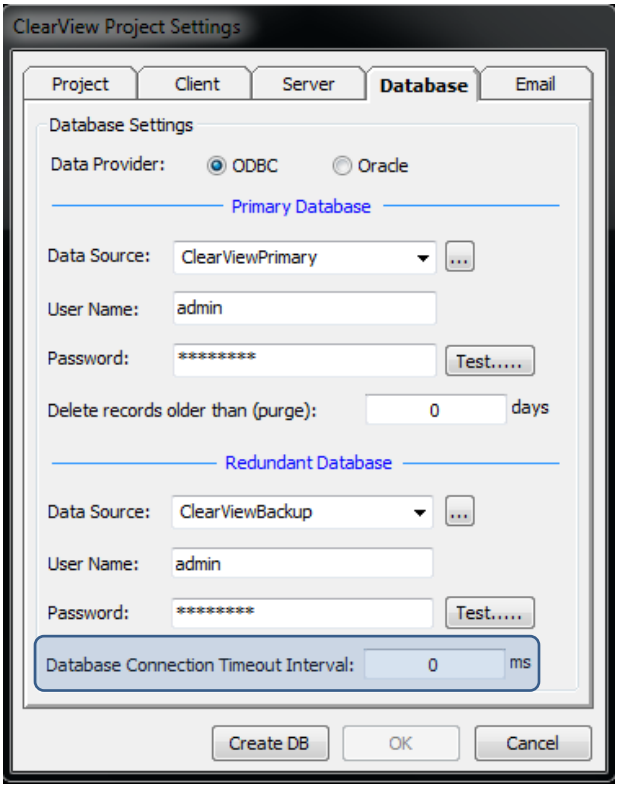


Figure #19.4

Database Connection Timeout Interval – specifies how often ClearView-SCADA Server would check database connectivity health.

Historical Data Replication between Databases

Historical data replication between databases is needed for the general configuration as the data is written by the active server only. During periods of server inactivity, the historical data for those periods will be missing in the server's database. This missing data will reside in the database of its peer. This complicates historical data analysis because it requires the merging of historical data from both databases. ClearView-SCADA database replication features automatic and seamless data merge. This feature is available for MS SQL Server databases only and is implemented as a number of

MS SQL Server objects. These objects are generated on top of the basic redundant configuration using a special utility that will be described later.

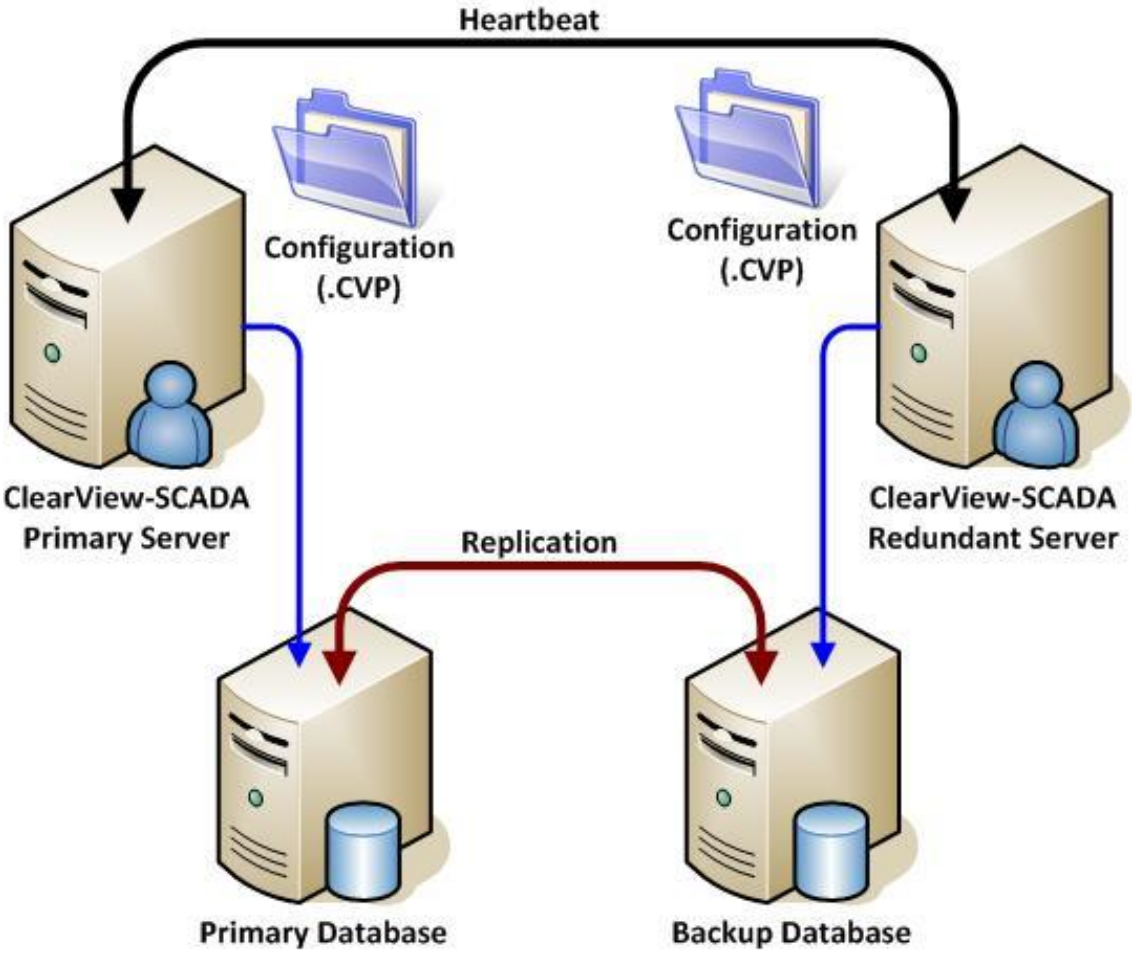


Figure # 19.5

Description of Steps for Data Replication

Primary and backup databases work absolutely symmetrically with historical data replication. The following SQL Server objects are created on each database.

Name	Type	Description
HistoryBuf	Table	The same as History table but having extra [id] column
EventLogBuf	Table	The same as EventLog table but having extra [id] column
LastReplicatedIDs	Table	This table has single record having columns for storing id of the last records from HistoryBuf and EventLogBuf tables, correspondingly, that were merged by the peer SQL server into its History and EventLog tables.

Name	Type	Description
tr_History	Trigger	Trigger for insert into History table. Duplicates inserted records in the HistoryBuf table
tr_EventLog	Trigger	Trigger for insert into EventLog table. Duplicates inserted records in the EventLogBuf table
sp_PurgeHistoryReplicationData	Stored Procedure	Deletes records from HistoryBuf table that were already merged by the peer into its History table.
sp_PurgeEventLogReplicationData	Stored Procedure	Deletes records from EventLogBuf table that were already merged by the peer into its EventLog table.
sp_PullHistoryReplicationData	Stored Procedure	Copies records from HistoryBuf table of the peer into this server's History table and purges copied records.
sp_PullEventLogReplicationData	Stored Procedure	Copies records from EventLogBuf table of the peer into this server's EventLog table and purges copied records.
job_PullDataFromPeer	Job	Once a minute calls sp_PullHistoryReplicationData and sp_PullEventLogReplicationData
RedundServer	Linked Server	Used by sp_PullHistoryReplicationData and sp_PullEventLogReplicationData to access objects on the peer SQL server.

Any time the records are inserted into the History or EventLog tables they are copied into corresponding buffer tables. The [id] column of these tables automatically acquires a sequence number of the record.

Once a minute the job_PullDataFromPeer is called. It selects from the peer server the records from the buffer tables with [id] larger then "last replicated id" taken from the LastReplicatedIDs table, inserts those records into the corresponding table, modifies LastReplicatedIDs table, and purges the buffer tables on the peer server.

Note: there is special provision in the triggers that prevents from duplication of the records that are inserted during job_PullDataFromPeer execution.

Note: SQL Server Agent must be started to make job_PullDataFromPeer work.

Configuring Replication using GUI

Historical data replication is set using Database Replication Configuration Console. Where is this console located?

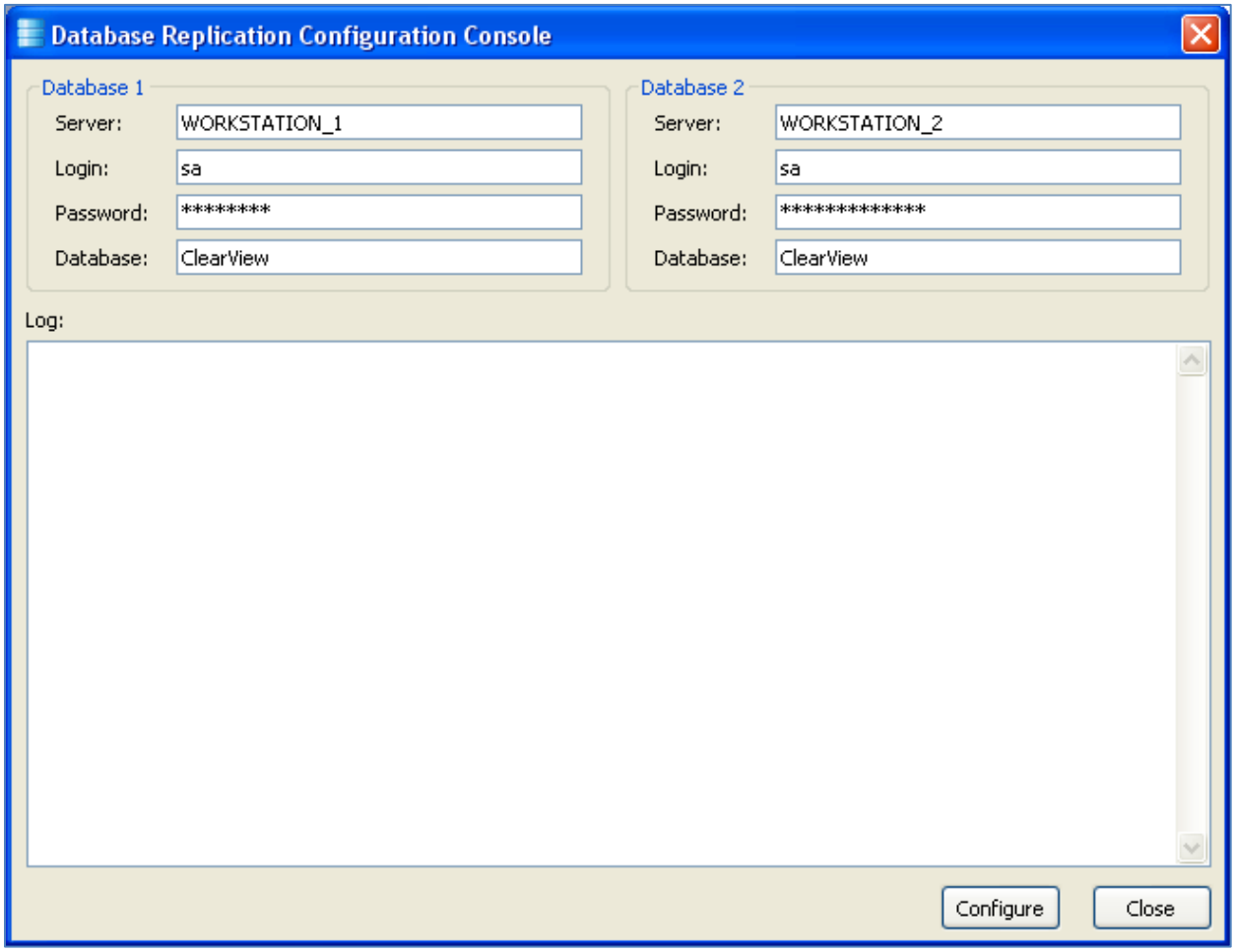


Figure # 19.6

The user needs to input the necessary information to connect the primary and backup database servers and then click **Configure**. The utility will create required objects on SQL Servers.

Note: the user must provide logins that have privileges sufficient for creating and/or modifying SQL Server objects described above.

Enterprise Consumer's Redundancy

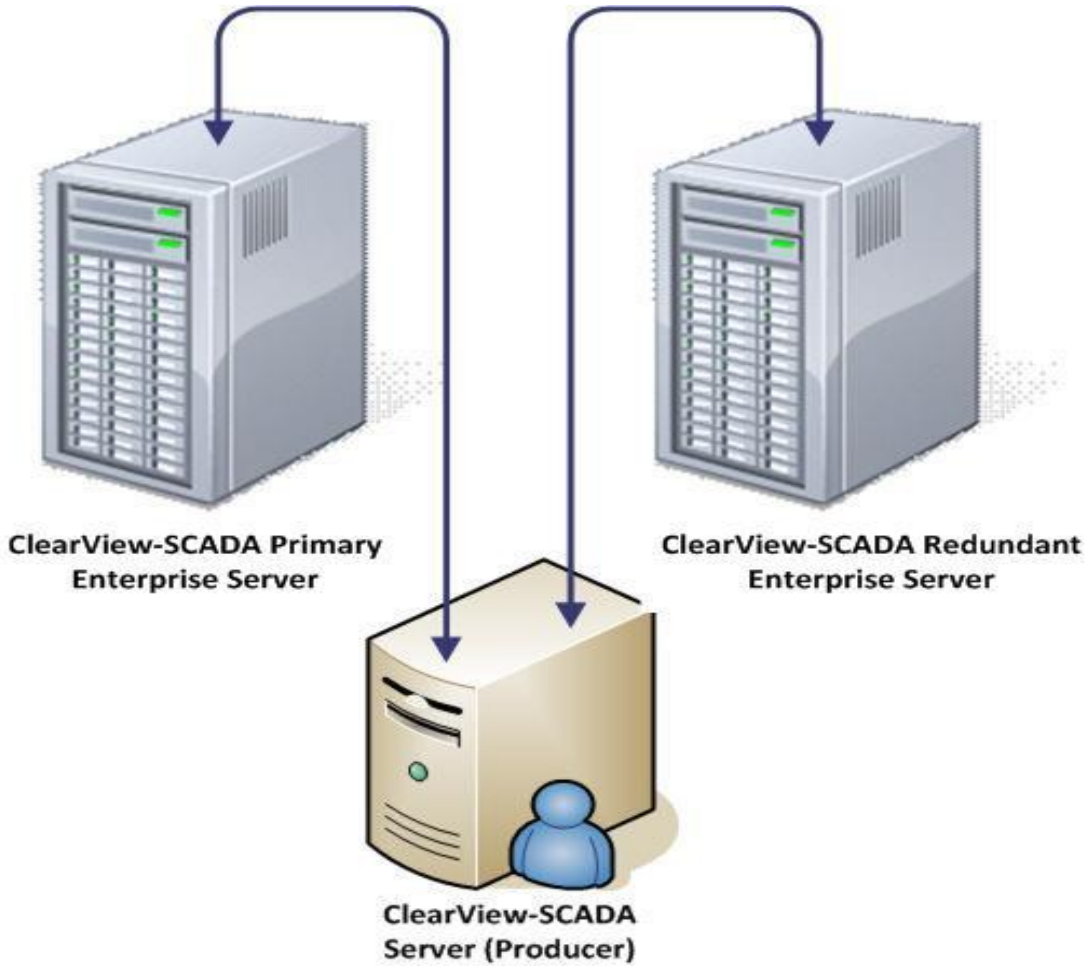


Figure #19.7

Description

Provided that there is a primary and a redundant Consumer, the Producer just needs to communicate with the two Consumers in parallel.

Configuration Settings

The options Host and BackupHost are used for making the Producer connect to primary and backup Consumers. These options are from ProducerAPI.cfg file located in the "Bin" folder of the ClearView-SCADA deployed on the Producer's machine. The options should be edited manually, and their values should be IP addresses of the corresponding hosts.

Enterprise Producer's Redundancy

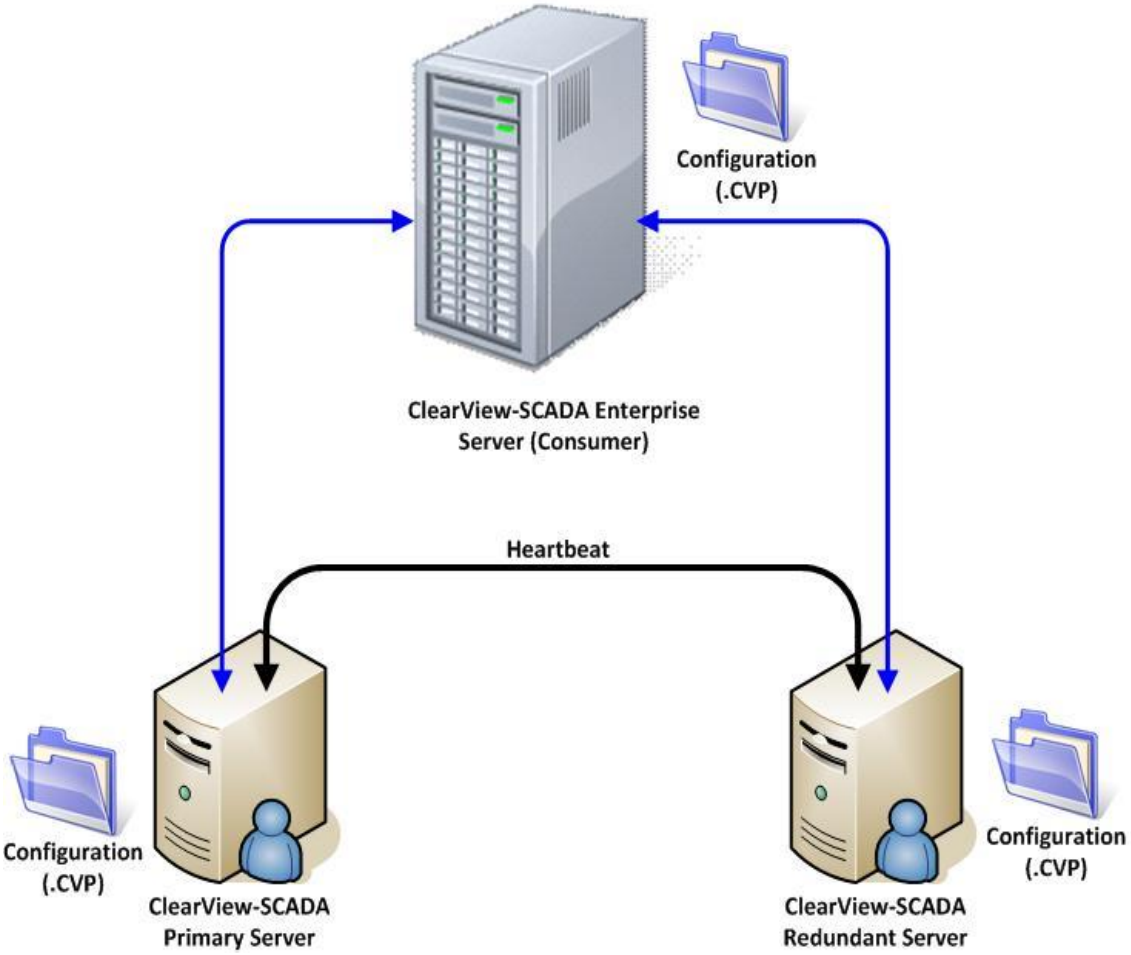


Figure #19.8

Description – Producer Redundancy

Producer redundancy for the Enterprise Server works on top of the basic ClearView-SCADA redundancy. The ClearView-SCADA server redundancy for producers is configured as previously described. Then the primary and backup servers are configured for connecting to the same consumer, this is a step required for Enterprise Producer redundancy. The consumer will have two sets of identical tags—one from the primary and another one from the backup Producer. Depending upon the way the customer's software manages producer redundancy, the inactive producers from the primary/backup pair may or may not communicate with the Consumer. For that option, SendUpdatesInBackupMode is used. It equals 1 by default; that means that both Producers send updates to the Consumer. If the option equals 0, then the inactive producer will not send updates, thereby reducing the load on the consumer. ClearView-SCADA provides its own implementation of the functionality behind the consumer. It relies on the naming conventions for item IDs of the producer tags. The item IDs have the following format:

Producers.<*Producer Name*>.$<*Short Item ID*>.

Here "Producers" is a hardcoded name, <*Producer Name*> is the configured name of the producer, and <*Short Item ID*> is generated by consumer from the tag name as used by the producer. For instance, if primary and backup producers have tag "SomeTag", then then consumer will show items with the IDs Producer.PrimProd.$SomeTag and Producer.BackProd.$SomeTag, where *PrimPrid* and *BackProd* are names configured for Primary and Backup producers.

<*Producer Name*> may be provided in the option Name located in the ProducerAPI.cfg file. If not provided, the producer uses computer name for producer name.

WARNING: One cannot guarantee that the computer name resolved by the producer on startup will stay the same among restarts because it depends upon an OS setup that may change over time. Hence, in order to provide a robust configuration, it is highly recommended to set producer names explicitly even if they coincide with the computer names.

If the client application behind the Consumer knows how names of the primary and backup Producers match, it may match corresponding tags. This way the ClearView-SCADA Server works behind the consumer. This is done in the following way.

The project for this server includes tags for the items of the primary producer only. For instance, for the "SomeTag", the above tags database should include the tag "SomeTag" with the ID Producer.PrimProd.$SomeTag.

In addition, the server uses a file that maps primary-backup producers. This file name is similar to <Project Name>.cvp.prod, is located in the same folder as the <Project Name>.cvp file, and contains the lines:

<Primary Producer Name>=<Backup Producer Name>

For Each Primary/Backup Producer Pair when ClearView-SCADA server opens a project, it checks for the value of the EnableProducerRedundancy option in the *.cvp file. If the value equals 1, it reads information from <Project Name>.cvp.prod file. Then for each tag in the project, it registers two items against the consumer OPC Server—one with the item ID in the tags database, and another with the item ID of the matching tag. In our example, "SomeTag" would have the item ID Producer.PrimProd.$SomeTag. In this case, the following two items would be registered: Producer.PrimProd.$SomeTag and Producer.BackProd.$SomeTag.

After that the updates from consumer OPC Server (write from the ClearView-SCADA Client) for any of the items Producer.PrimProd.$SomeTag and Producer.BackProd.$SomeTag are considered by CV Server as updates of the "SomeTag". Symmetrically, the writes for "SomeTag" will trigger writes for both items.

In this way, the ClearView-SCADA Client connected to ClearView-SCADA Enterprise Server will be aware of a single tag; the fact that it is duplicated in two producers will not be noticed.

Recommended Configuration

1. Configure basic redundancy for Producer ClearView Servers
2. In ProducerAPI.cfg files of each producer set the value for option Name equal to the name of the corresponding computer

3. In the project file (cvp) used by each producer set option SendUpdatesInBackupMode equal 0 and EnableProducerRedundancy equal 1. Or use ClearView-SCADA project settings GUI interface form

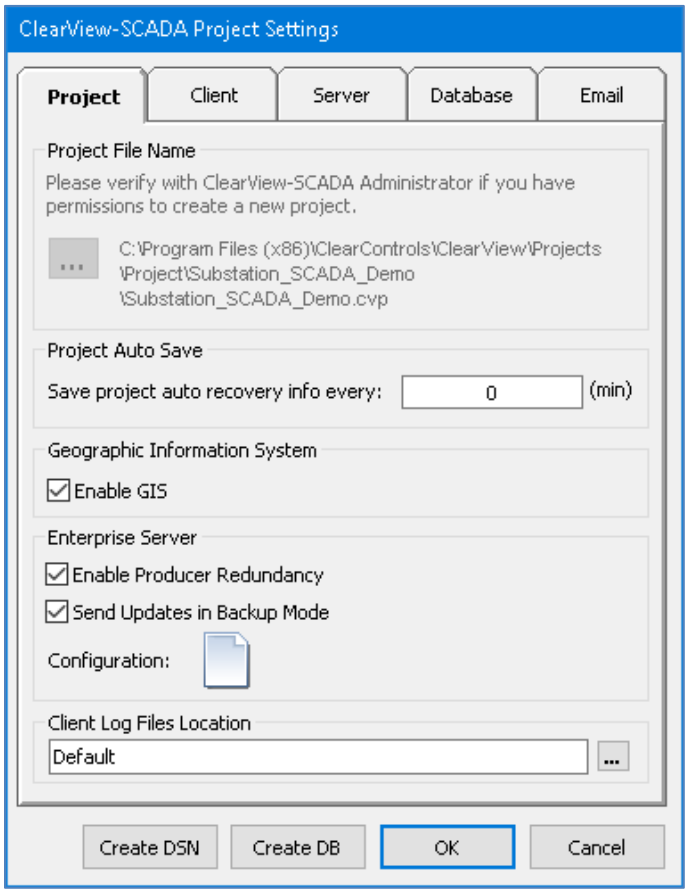


Figure #19.9

4. Start all producers.
5. Using OPC Console create configuration for consumer OPC Server making sure that items from both primary and backup producers are included and the set of items for primary producer coincide with the set of items of its peer.
6. Create project for ClearView-SCADA Enterprise Server using ClearView-SCADA client. Put checks on Project page like shown above. The rest of project setting do not influence producer redundancy.

7. On the Tag Database screen create required tags for items from primary producer(s) only.

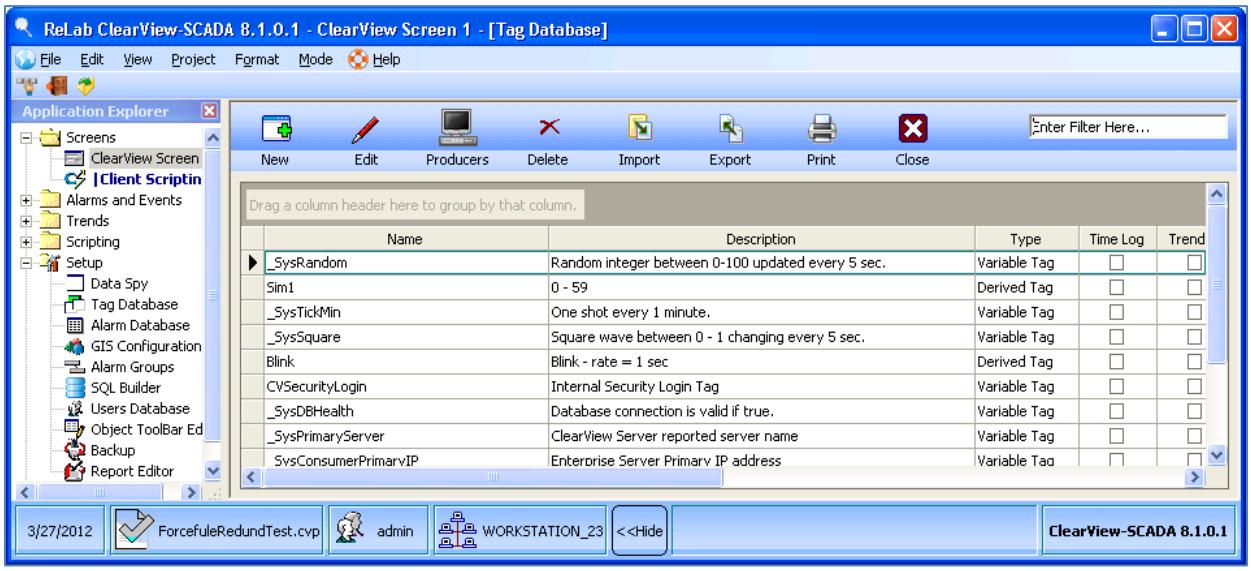


Figure # 19.10

8. Click **Producers**.
9. On the form that appears, define **Primary/Backup** producer pairs.
10. Save the project.

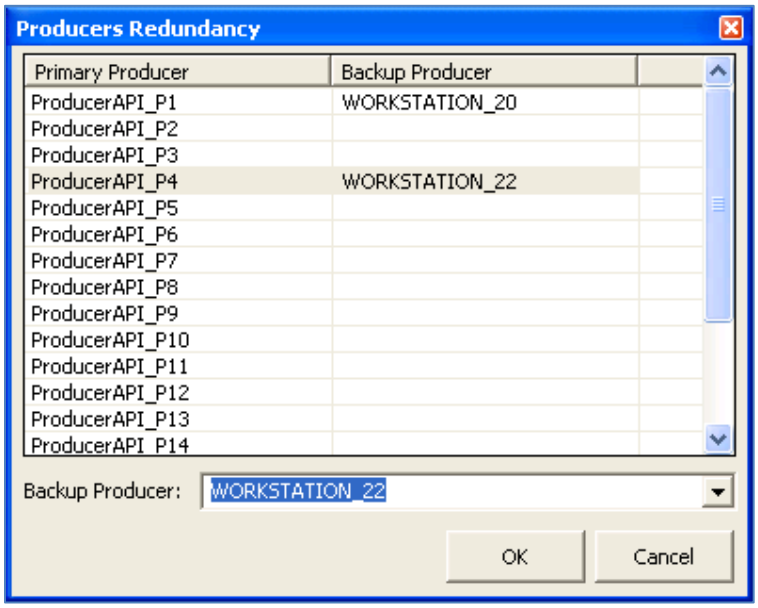


Figure # 19.11

ClearView-SCADA Redundancy topology

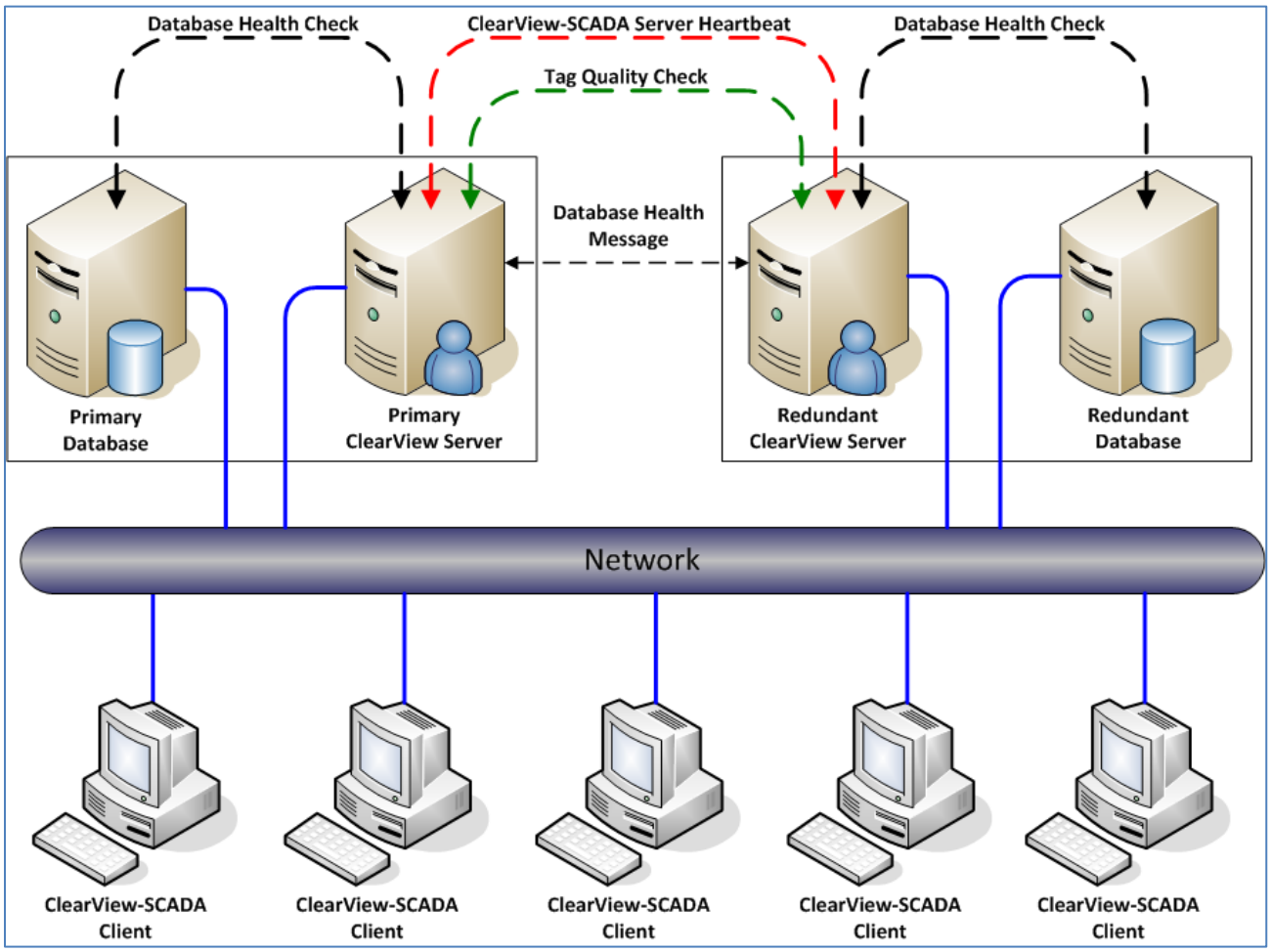


Figure #19.12

SECTION 20

Enterprise ClearView Server

Registering ClearView-SCADA Server (Enterprise Producer)

1. Open ClearView-SCADA Server Interface
2. Navigate to Main Toolbar > Settings > Register

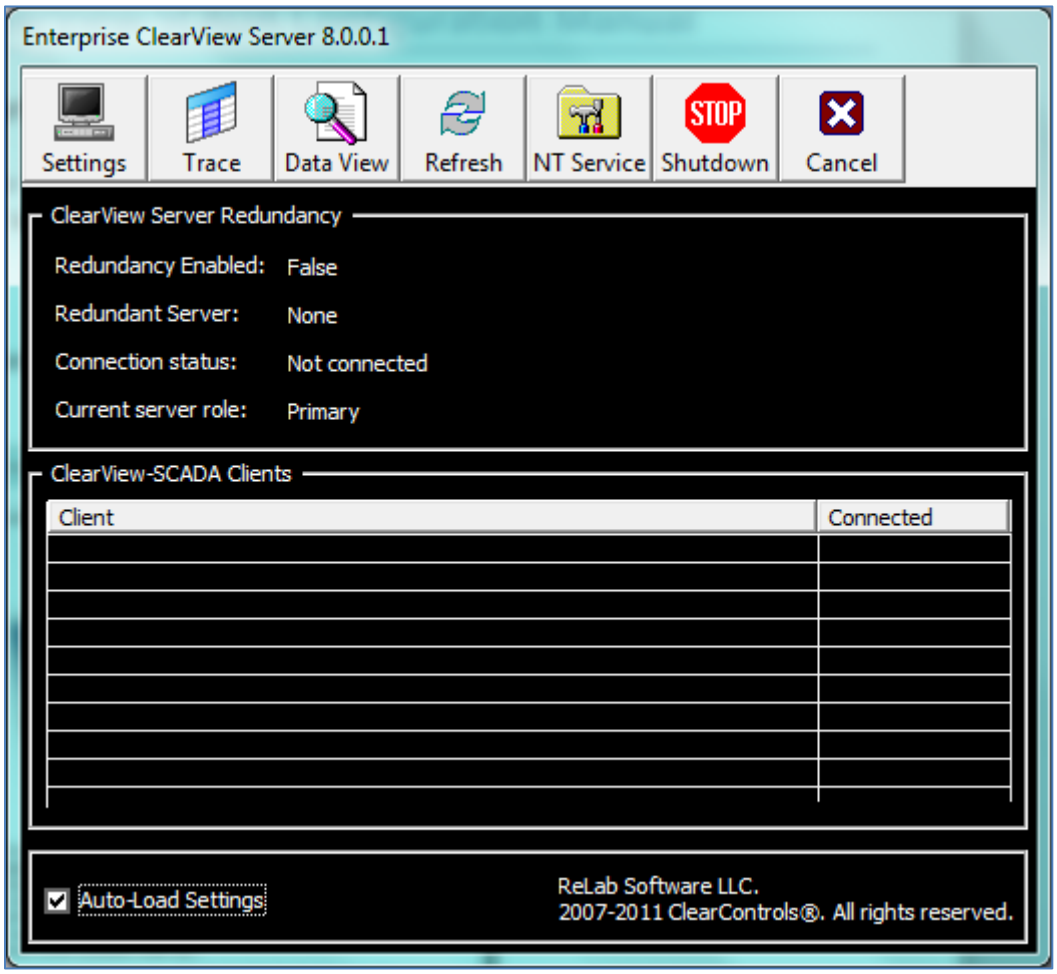


Figure # 20.1

1. Check ClearView Enterprise Server checkbox.
2. Contact ReLab Software and provide the **Customer ID** number displayed.

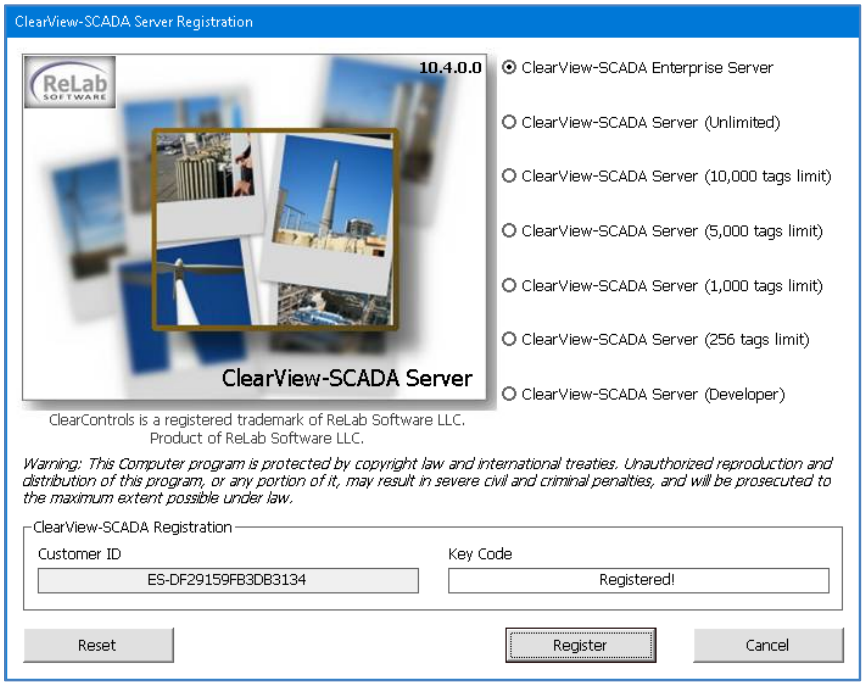


Figure #20.2

4. Enter the **Key Code** provided by ReLab Software
5. Click **Register** button
6. Restart ClearView-SCADA server

Configuring ClearView-SCADA Server (Enterprise Producer)

1. Open ClearView-SCADA Server Interface
2. Navigate to **Main Toolbar > Settings > Configure**

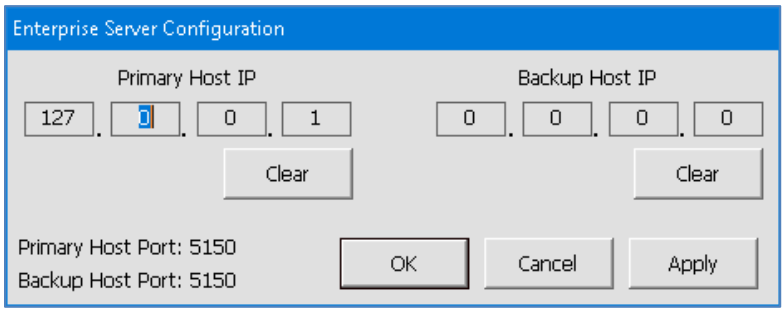


Figure #20.3

3. Enter Primary and Backup host IP addresses for ClearView-SCADA Consumer
4. Click **Apply** and then **OK**.
5. To clear the current IP addresses, click **Clear**.

Registering ReLab Enterprise Consumer

Install the latest version of ReLab Software OPC Product Suite

1. Open the ReLab OPC Console.
2. Navigate to **Main Menu > Tools > Register Enterprise**.

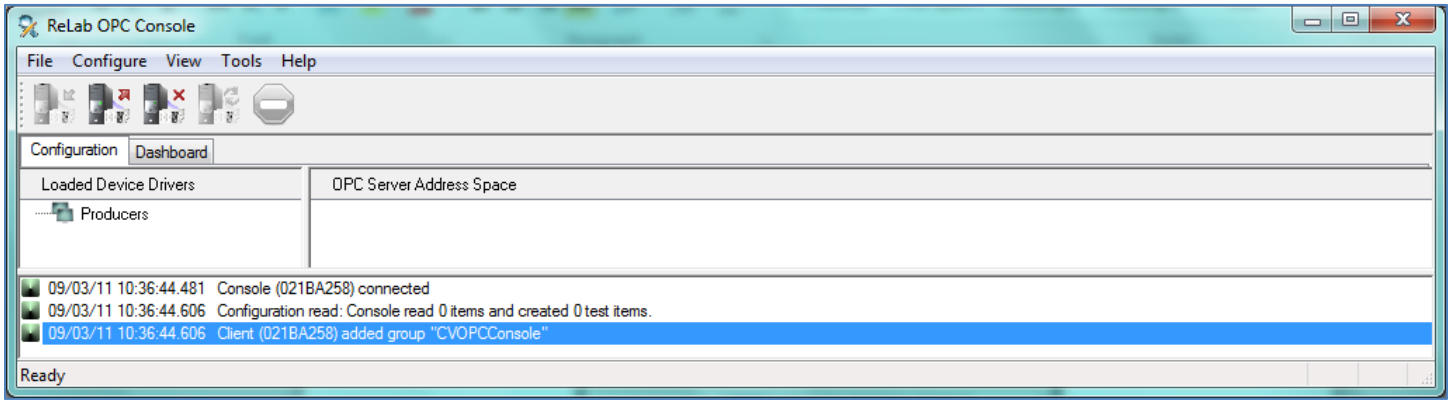


Figure #20.4

3. Copy the code and forward the customer code to ReLab Software
4. Enter the license code provided by ReLab Software
5. Click **Register** button
6. Restart OPC Server

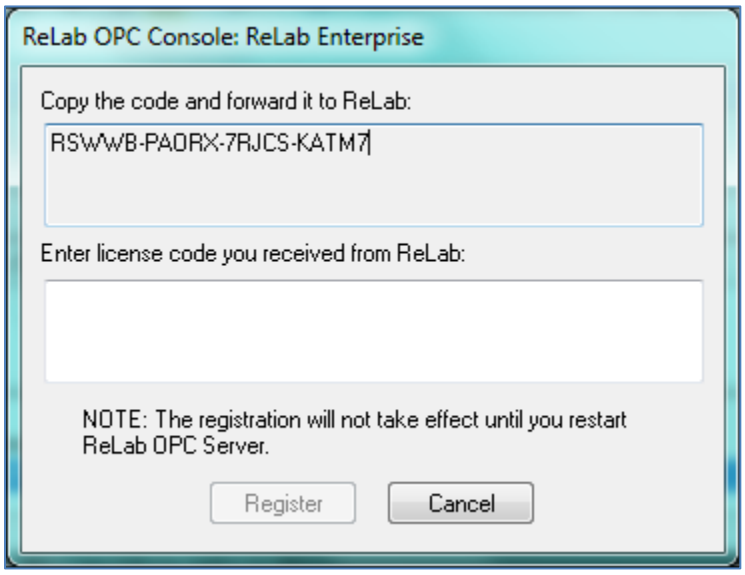


Figure #20.5

Configuring ReLab Enterprise Consumer

Make sure that the ClearView Servers Producer(s) are configured and started

1. Right click Producers folder and select **Reload** menu item

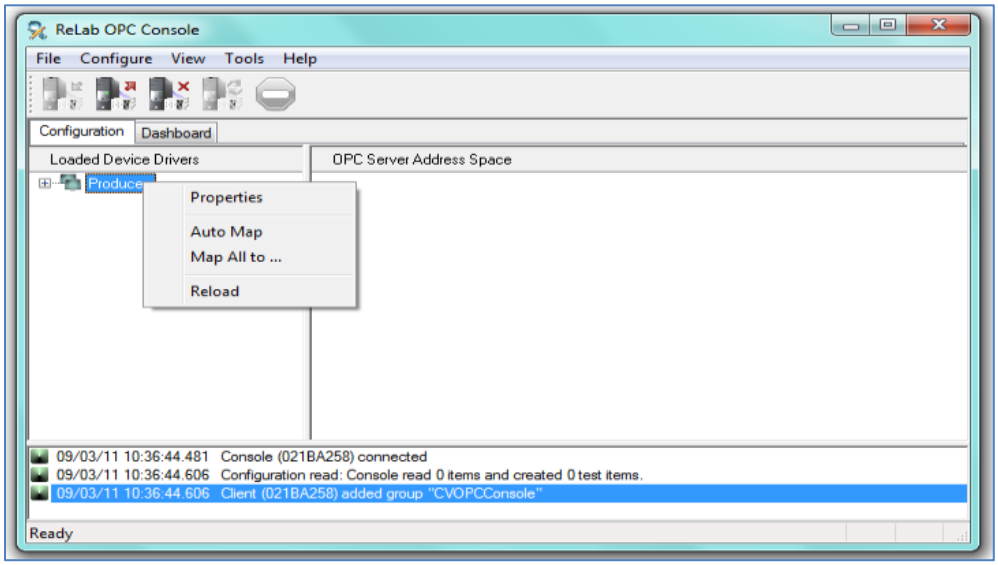


Figure #20.6

2. Expand all folders (Producer, OPC, Derived, Variable and Alarm) and map required items
3. Save Configuration

Now all of the Producers tags and alarms are available to any connected OPC Client (ClearView –SCADA Enterprise)

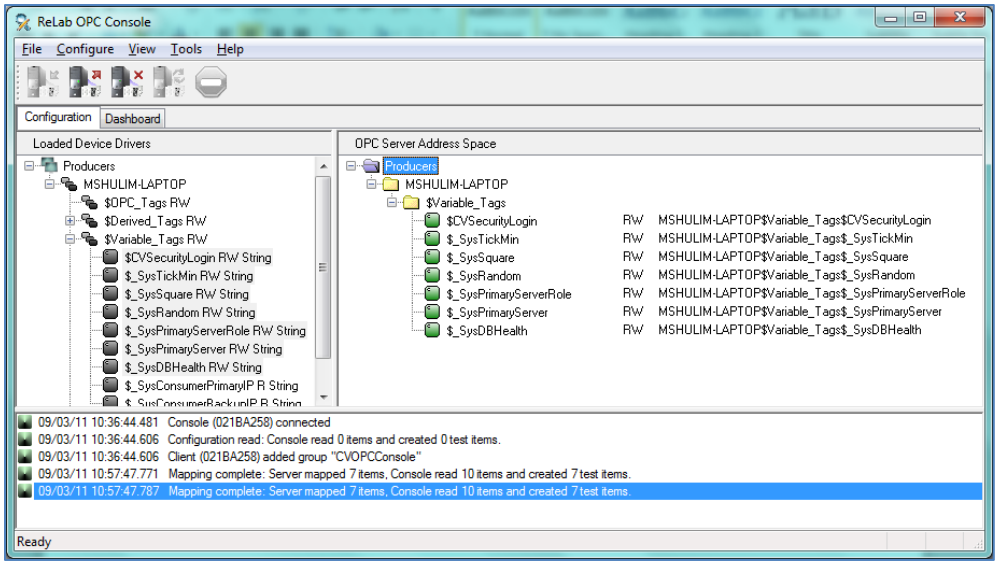


Figure #20.7

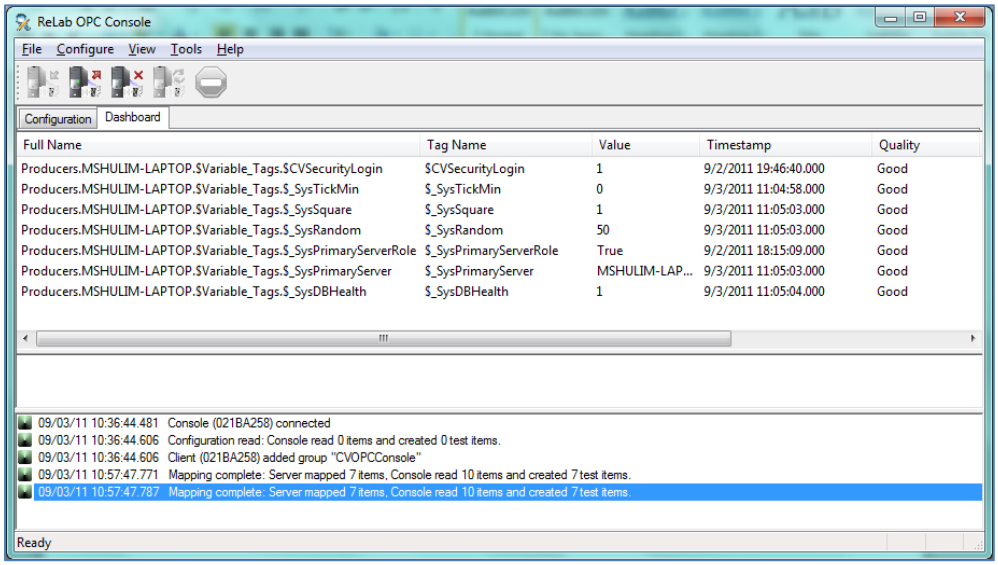


Figure #20.8

SECTION 21

SQL Server Backup and Restore

Registering SQL Server Database Backup and Restore Application

How to register your software

1. Start Database Backup/Restore Configuration Console
2. Retrieve **Customer ID** number
3. Contact ReLab Software and provide the **Customer ID** number
4. Enter the **Key Code** provided by ReLab Software
5. Click **Register button**

Figure #21.1

SQL Server Database Backup & Restore Configuration Console

After successful registration the SQL Server Database Backup and Restore configuration console will appear.

Figure #21.2

NT Service

The ClearView-SCADA server requires that you start the SQL Server Database Backup and Restore application as an NT Service. To do this, perform the following steps:

1. Click the NT Service icon

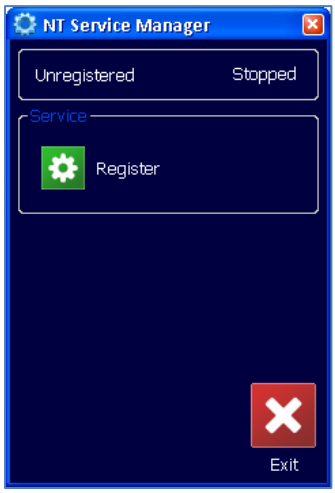


Figure #21.3

2. Click **Register**. The message box will appear confirming that the service is has been registered.

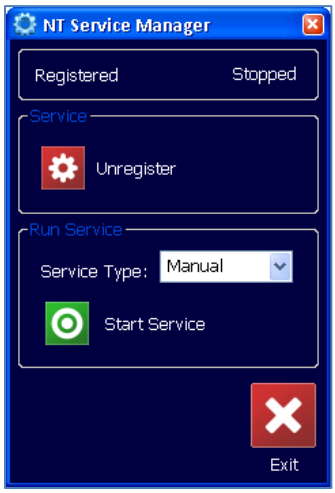


Figure #21.4

3. Select **Service Type- Automatic** from the drop-down list, and then click **Start Service**. A message box will appear confirming that the service is has been started.
4. Click Exit to return to the Backup and Restore configuration console.

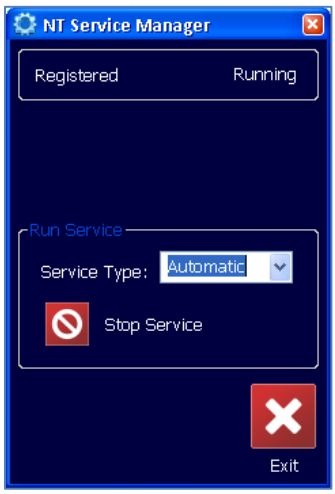


Figure #21.5

Tip: To ensure that the SQL Server database starts before SQL Server Backup/Restore Service "CVB" Service Make sure that service dependencies are configured:

- Stop "CVB" Service
- Open command line (CMD.exe) interface as Administrator
- Enter the following line to set dependencies: **sc config CVB depend= "SQL Service Name"**
- If multiple dependencies are required the "/" separator can be used (example: sc config ServiceA depend= ServiceB/ServiceC/ServiceD)

Configuring Databases

Click **Databases** in the SQL Server Database Backup and Restore configuration console to configure SQL Server databases. You will need to authenticate yourself to SQL Server database before proceeding with database configuration. The interface provides two options depending on which primary database has been selected:

- Configure the existing database
- Create a new database

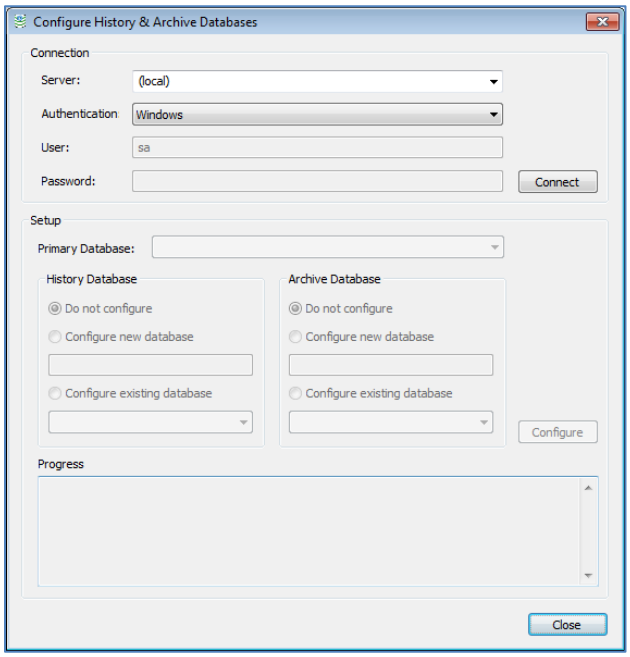


Figure #21.6

Configuring Data Backup & Restore

1. Specify when the application should perform a daily backup (hour/min).
2. Specify where data should be stored (path).
3. Specify where the data is located (Backup From). You will need to authenticate yourself to SQL Server database before proceeding with data backup and restore configuration.
4. Specify where the data will be restored to (Restore to). You will need to authenticate yourself to SQL Server database before proceeding with data backup and restore configuration.
5. Select which tables need to be backed up (Tables). The Event_Log table stores data for alarm and events and the History table stores data for all historical data.

   Note: Click **Refresh** to refresh the table list.

6. Click **Apply** or **OK** to create the configuration.
7. Click **Close** to return to the main configuration console.

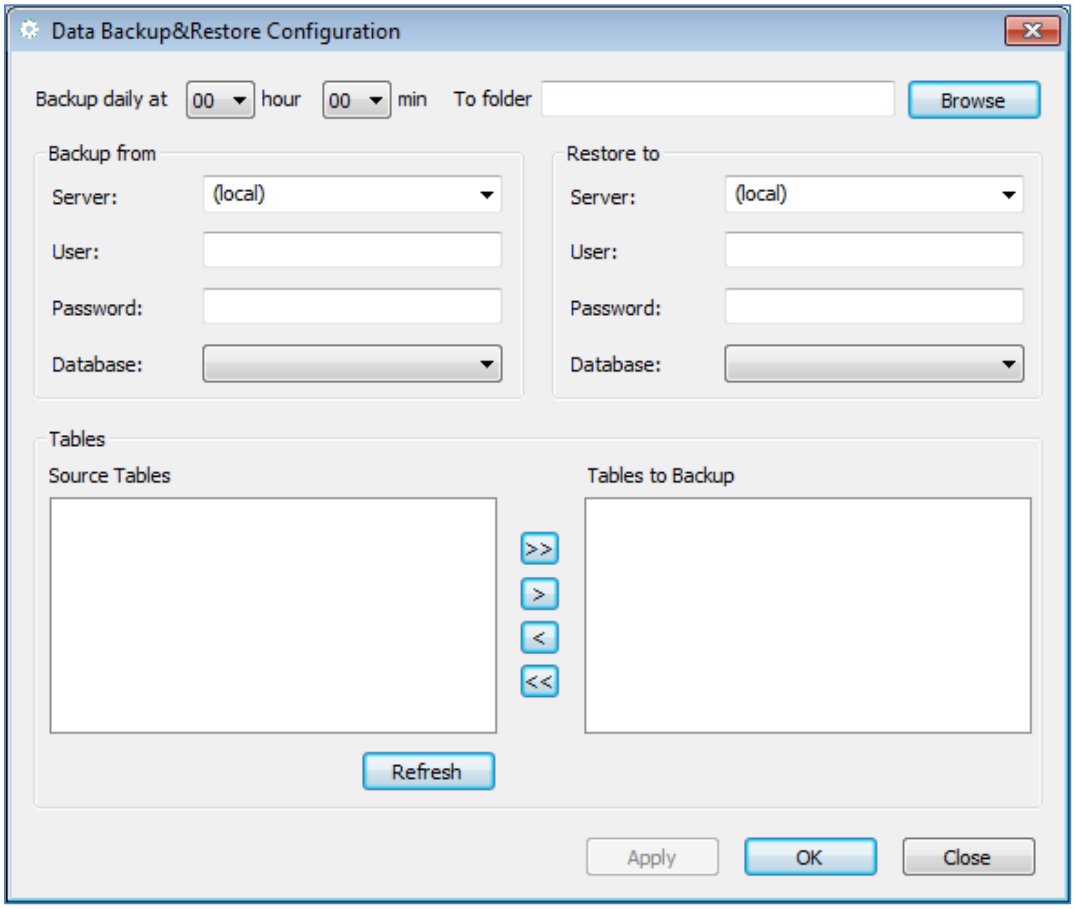


Figure #21.7

Restoring Data

To restore data, specify day or days of the data that needs to restored, then click **Restore**.

The data will be restored to the database that was specified above.

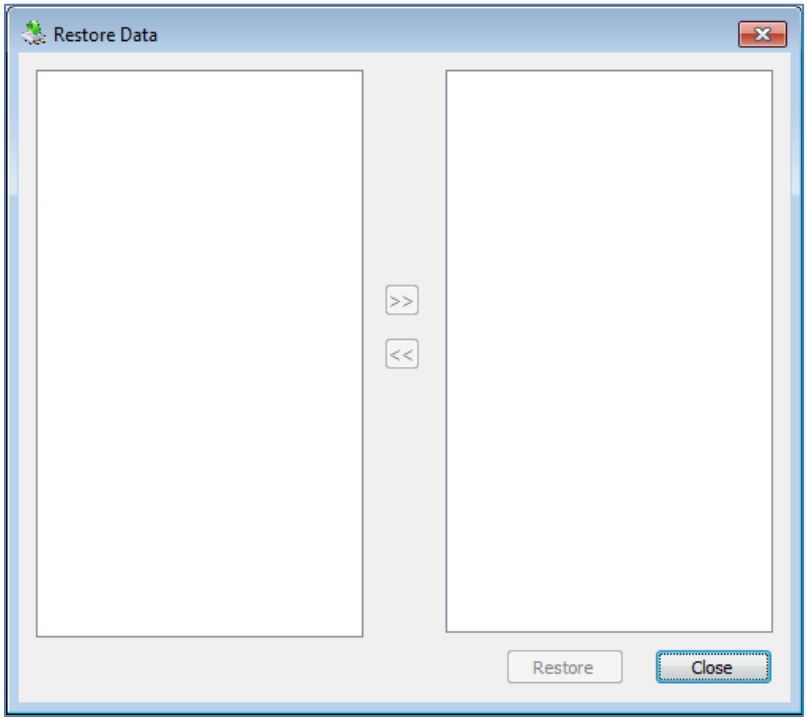


Figure #21.8

Purge History Database Tables

Backup and Restore Application can also be used to purge unnecessary data from the Historical database.

WARNING: Use caution when executing the Purge procedure.

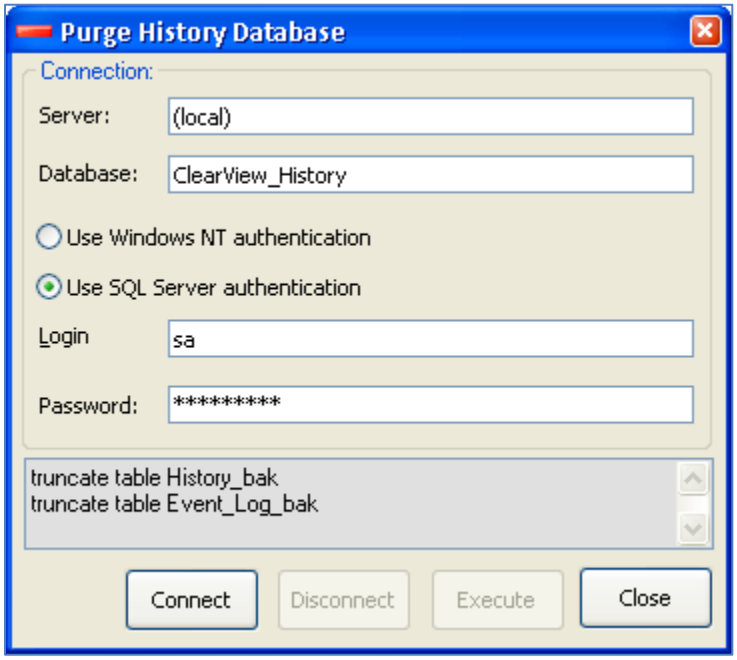


Figure #21.9

Connect – connects to the database.

Disconnect – disconnects from the database.

Execute – Executes Purge procedure.

Close – closes the application.

SECTION 22

ClearView-SCADA Object Animation Interface

ClearView-SCADA introduces new expression driven object animation module which allows for example dynamic coloring of objects such as power lines, switches, breakers, etc. The new feature is capable of coloring and animating very complex systems such as multi-source topology. Internal logic execution provides high performance and configurability in SCADA graphics animation.

The idea of ClearView Animation is to provide a User Interface to configure logical conditions that will trigger changes of an object's control items that in turn can be used in ClearView Wizard to change other object properties like Foreground Color, Background Color, line type, thickness, etc.

Configuration

Adding Objects (Controls) to Animation Configuration

Objects can be added only to a currently active screen

1. To make a screen active, double-click it in the Application Explorer.
2. Put selected screen in **Design** mode.
3. Right-click selected screen and select **List Controls (Figure #22.1).**

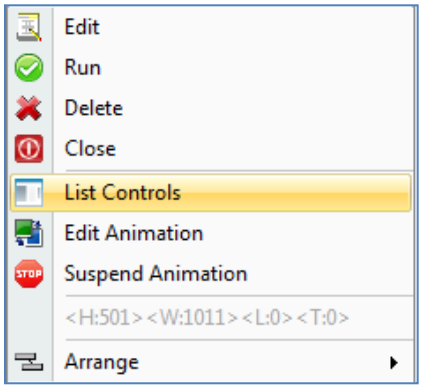


Figure # 22.1

Two methods are available to add objects to animation configuration console. Added objects will appear in **Controls** list.

- A user can select required objects from dropdown of List Controls (Shift + Mouse Click) (Figure #22.3)
- A user can select objects on the screen (Shift + Mouse Click, Figure #22.4)

Note: Single click without holding the Shift key would select a single object

- Right-click selected items in List Controls to display a context menu as shown on Figure #22.2

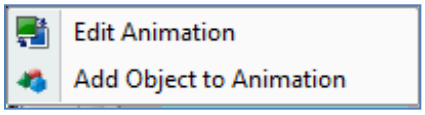


Figure # 22.2

- **Edit Animation** – Opens Animation Editor

- **Add Object(s)** – Adds object references to the Animation Editor and opens Animation Editor (Figure #22.5)

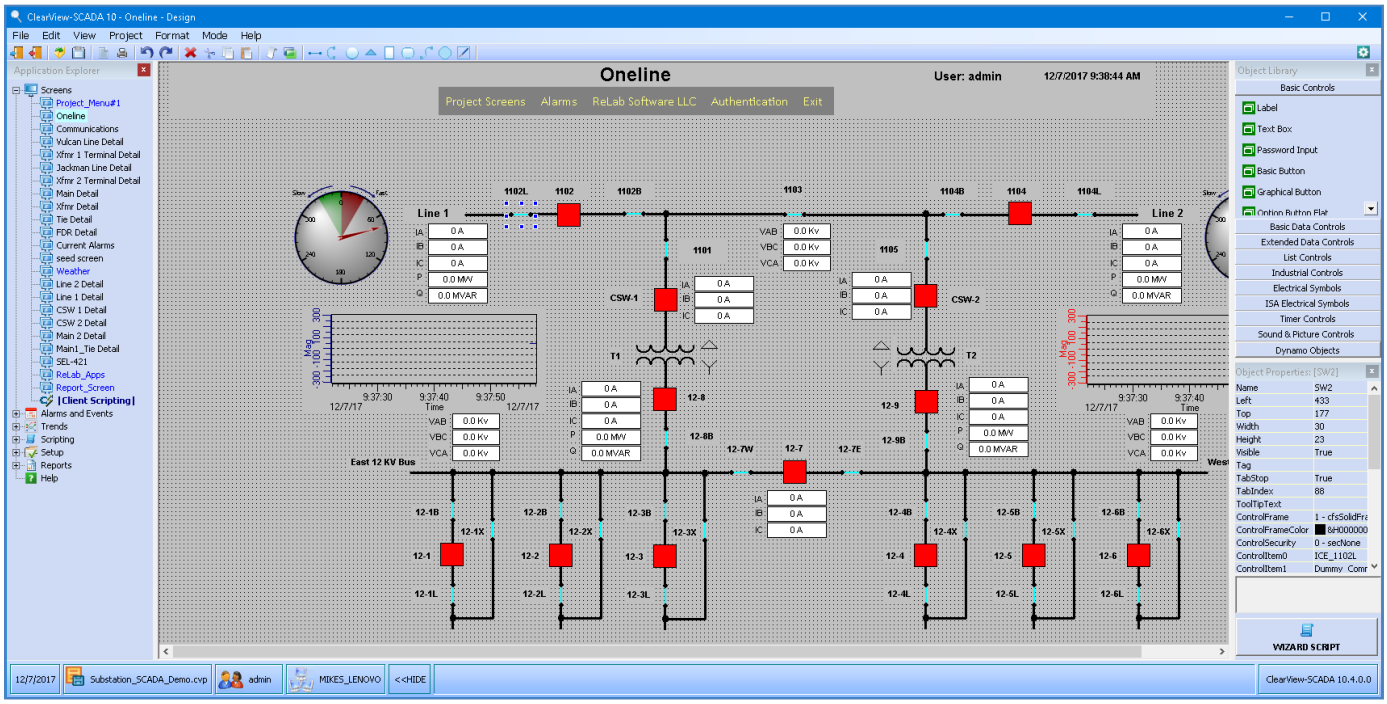


Figure # 22.3

Right clicking selected items on the screen will display a context menu as shown on Figure #22.4.

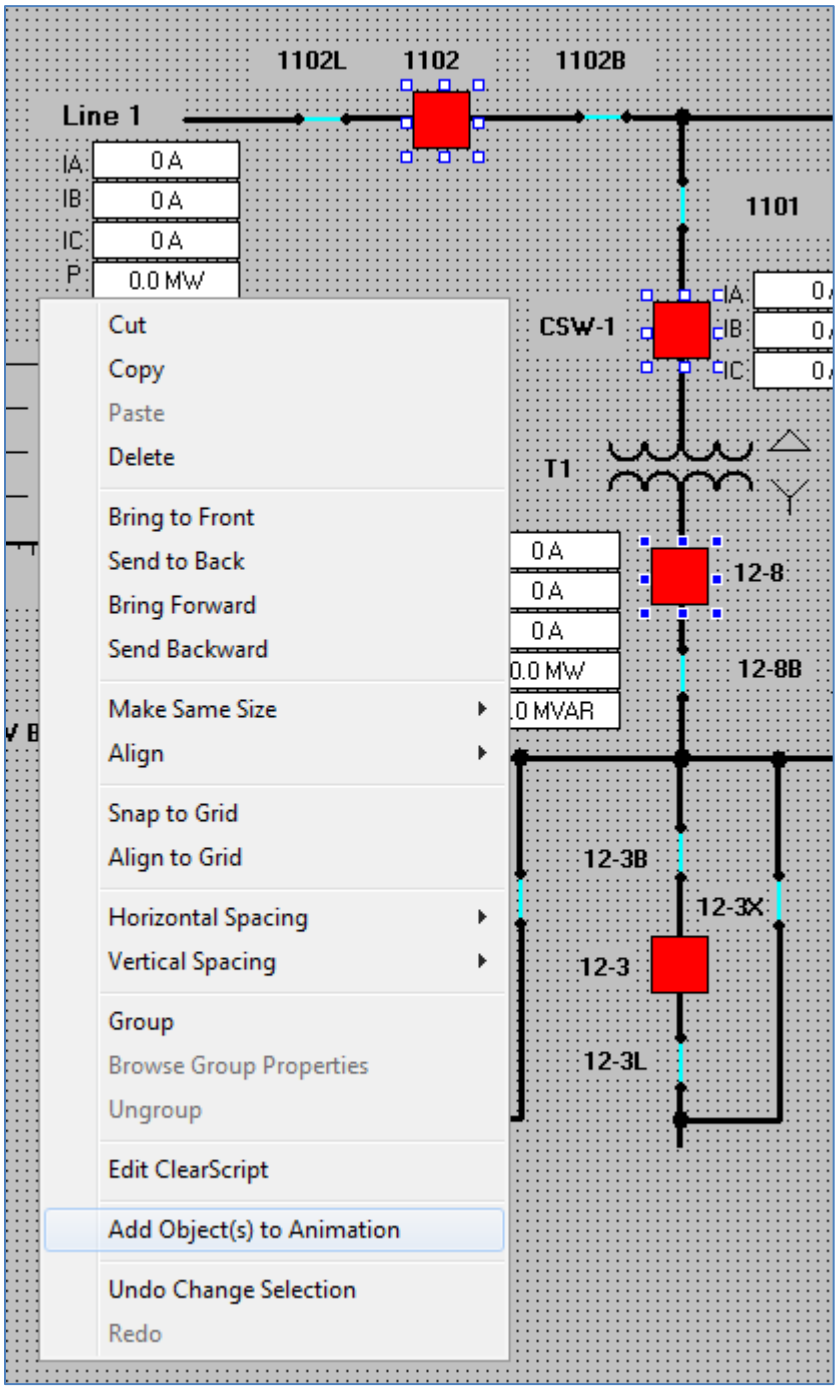


Figure # 22.4

Configuring Animation – Animation Editor

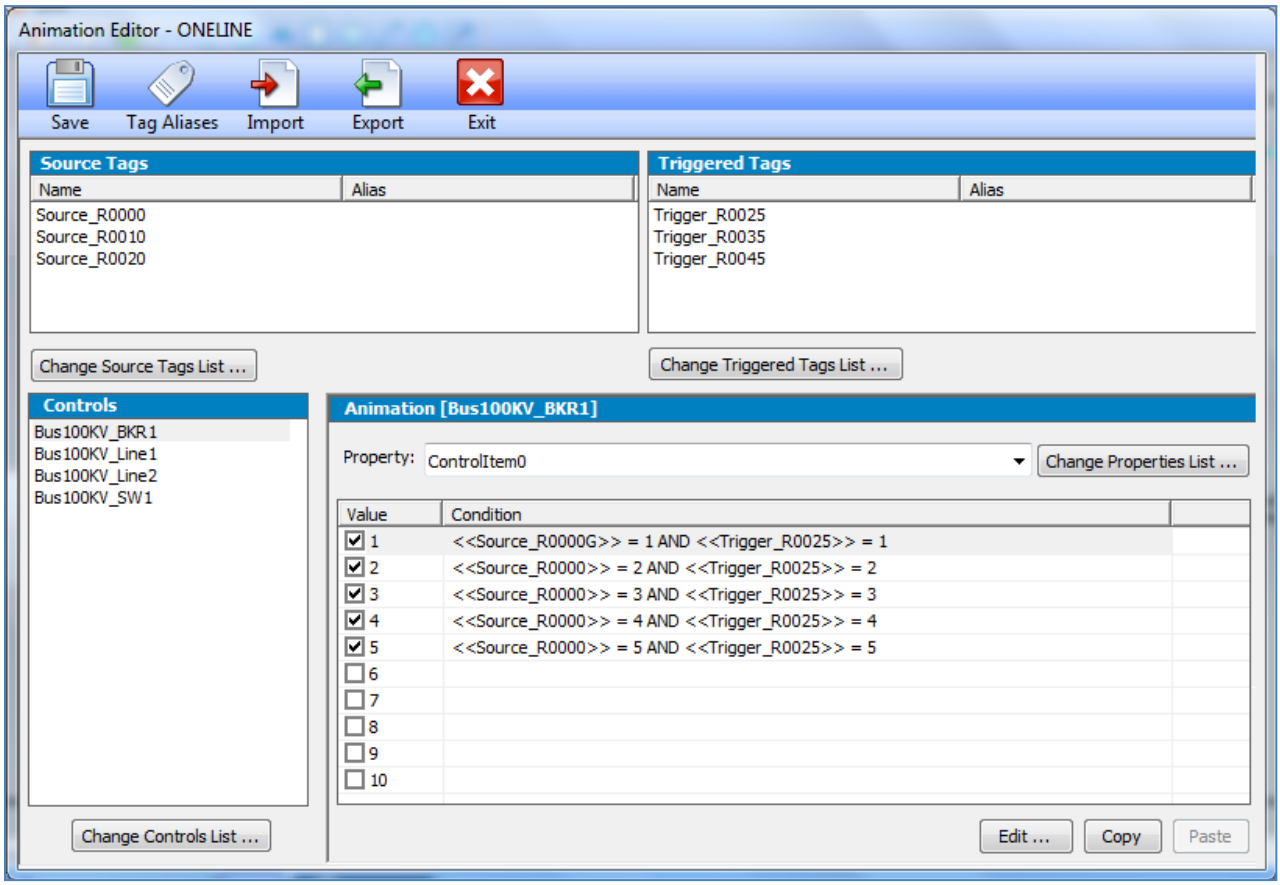


Figure #22.5

Changing Controls List

The user can add or remove objects in the configuration by clicking **Change Controls List…** in the Animation Editor window (Figure #22.5), which opens the Select Controls window (Figure #22.6)

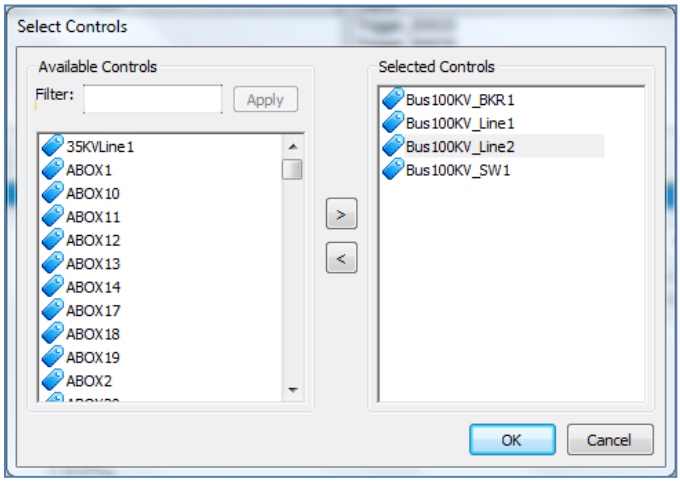


Figure #22.6

Changing Tags Lists

For the purpose of consistency, performance, and simplicity, the Animation Editor uses a pre-configured list of tags and aliases. When configuring conditions, the user can choose the tags from drop-down list and can use either tag name or tag alias.

To configure (add or remove) **Source** and **Trigger** tags, click either **Change Source Tags List**… or **Change Trigger Tags List…** in Animation Editor window (Figure #22.5)

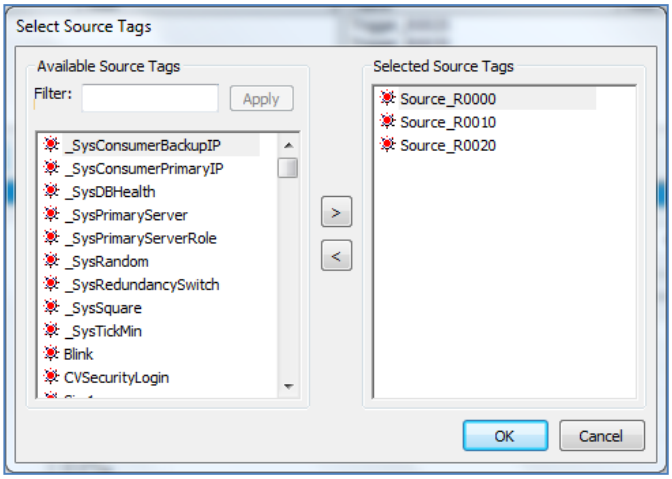


Figure #22.7

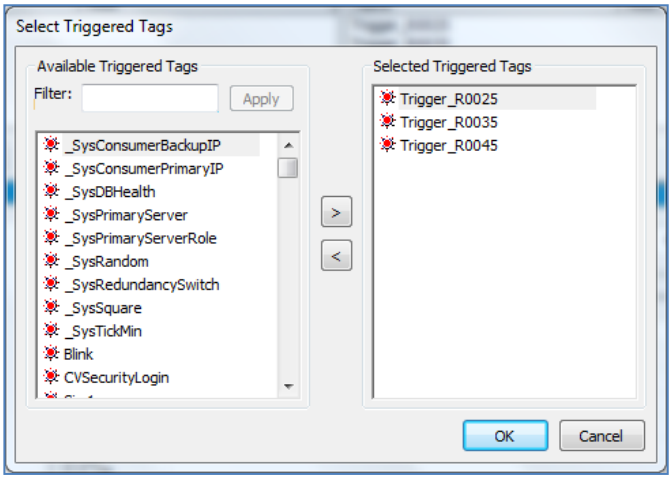


Figure #22.8

Note: The list in Source Tags and Triggered Tags are used to optimize animation performance; therefore, it is recommended that you use only the tags that are necessary for the Animation and will be used in Conditions.

Changing Tags Aliases

To assign Alias names to **Source** and **Trigger** tags, click **Tag Aliases** in the Animation Editor (Figure #22.5). Window for Tag Aliases will be opened (Figure #22.9).

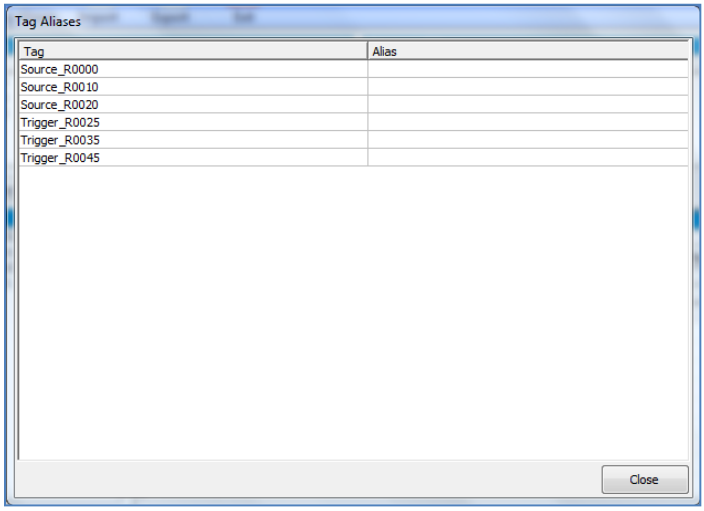


Figure #22.9

Note: Because aliases can be used instead of tags, the Alias name should not be the same as another tag name. Also, the Animation Editor will prevent you from having an Alias name the same as another tag name and will generate an Error Message.

Changing Properties Lists

With ClearView Animation, you can configure conditions that will change the values of special object properties called Control Items. There is a maximum of 10 Control Items per object named, from ControlItem0 to ControlItem10.

In Animation Editor window click **Change Property List…** to open a Select Properties window, where you can select or remove ControlItem(s) property values would be written (Figure #22.10). You can select one or more ControlItems at a time.

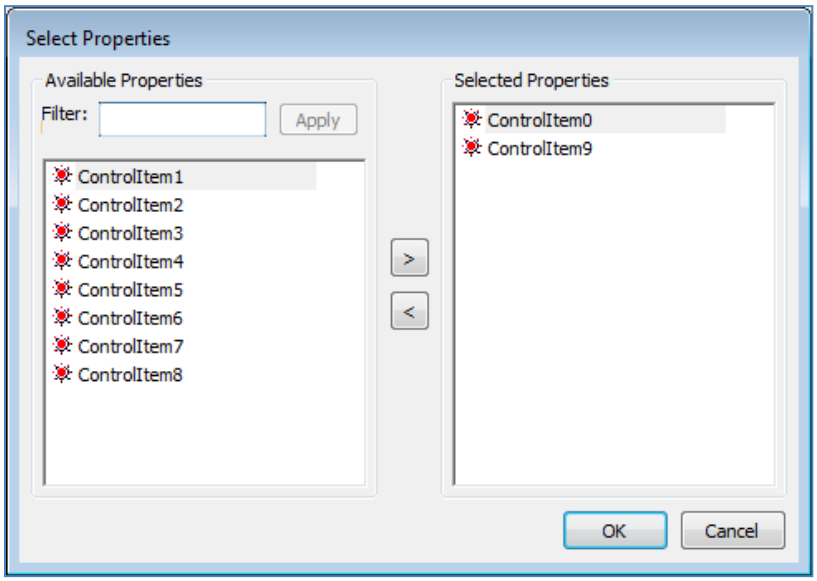


Figure # 22.10

Each ControlItem property has 10 states. Each state is triggered by a corresponding Condition. If the first Condition is evaluated to as TRUE, then the value of 1 will be assigned to the ControlItem. If the second Condition is TRUE, then the value of 2 will be assigned to the ControlItem, and so on until 10. Note that the Conditions are evaluated in order, from one 1 to 10. The first TRUE condition triggers an assignment of corresponding values to the Control Item and further evaluation stops.

Note: It is not necessary to enter all 10 conditions.

See also information provided in the tables on Table 22.1 and Table 22.2. The tables show the returned value of control property in abnormal cases.

Editing Expression (Expression Builder)

Two drop-down lists allow you to choose the **Source** and **Trigger** tags. Clicking **Insert** adds the selected tag to the expression interface embraced by the <<….>> identifiers.

Double-clicking a specific row or **Edit** (see Figure #22.5) will open Expression Builder interface (Figure #22.11).

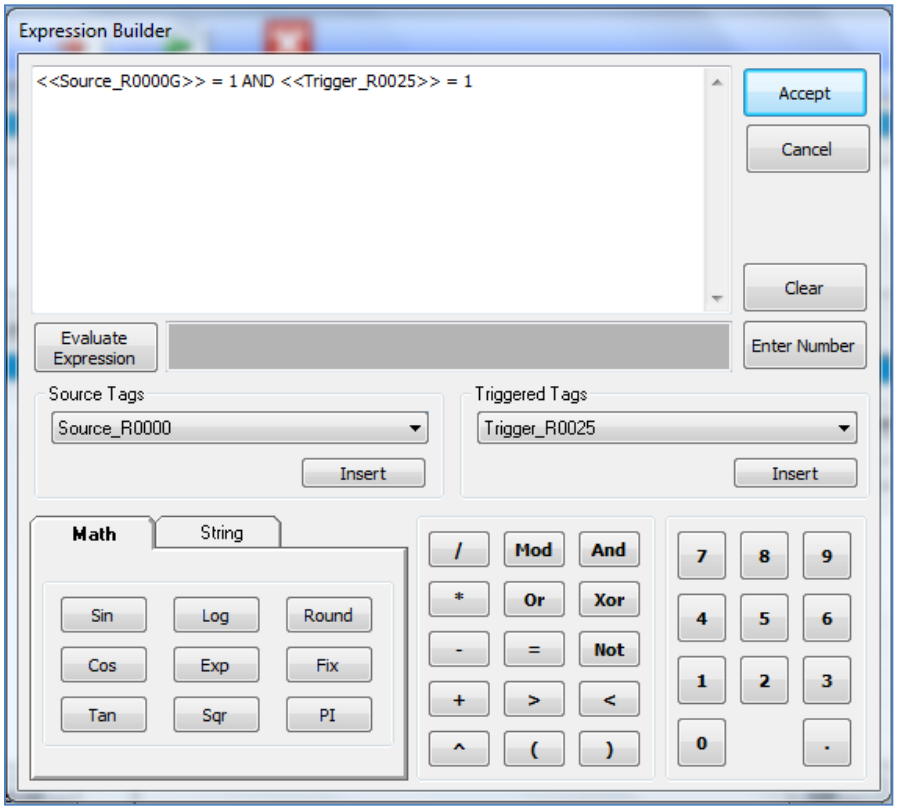


Figure #22.11

User has an ability to evaluate an expression before submitting it. Clicking **Evaluate Expression** opens the Tags Value Simulation interface (Figure #22.12). The user can enter any values for a particular tag and evaluate an expression by clicking **Evaluate**.

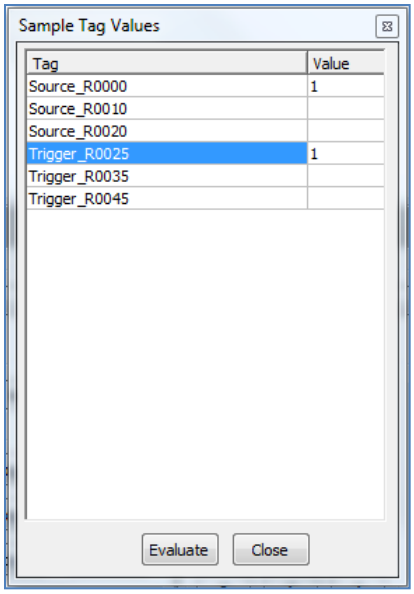


Figure #22.12

Other functions inside Expression Builder window (Figure #22.11) include:

- **Accept** – accepts configured expression and closes Expression Builder

- **Cancel** – closes Expression Builder without saving configured expression
- **Clear** – clears expression
- **Enter Number** – allows the user to enter a numeric value for Expression Builder in a separate window (Figure #22.13).

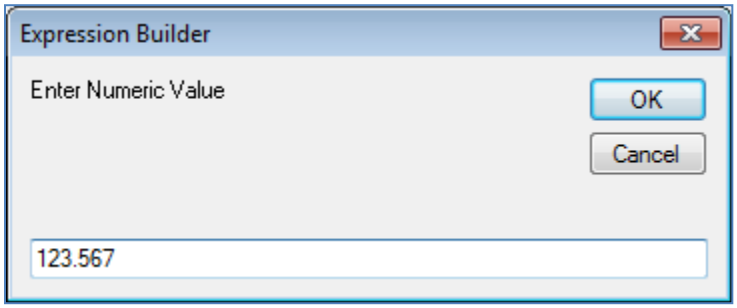


Figure #22.13

- **Operators (multiple buttons)** – any mathematical or logical function can be added by utilizing the Expression Builder interface. Although the operators and tags can be added manually, it is recommended that you use the drop-down lists and buttons.

Copying Expressions

ClearView Animation Editor allows you to Copy and Paste conditions. A condition can be copied by using Copy and Paste buttons at the bottom of the screen or by using context menus. The last can be invoked by right-clicking the mouse (Figure #22.14).

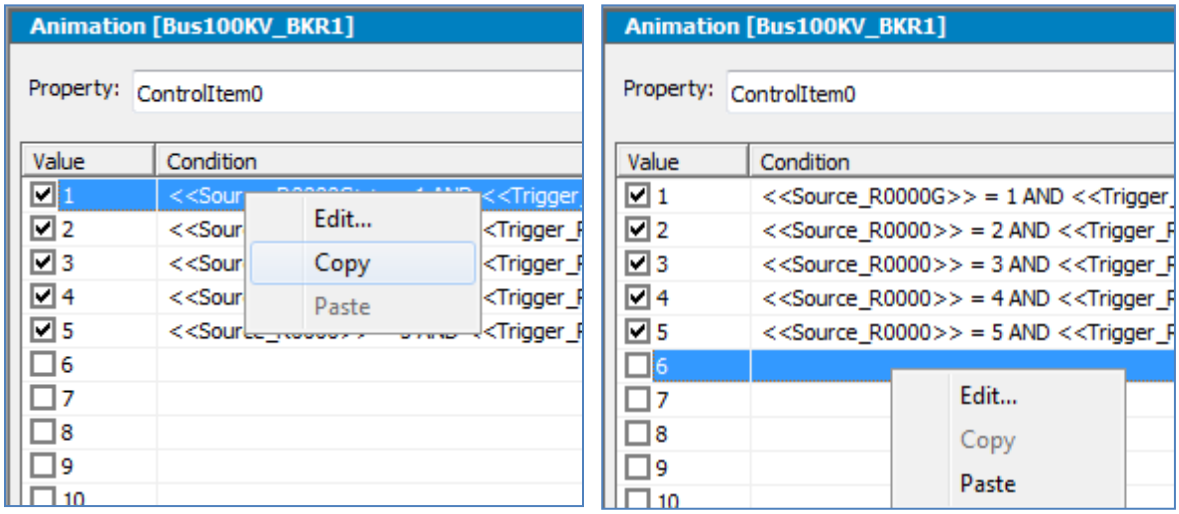


Figure #22.14

Copying a Control Animation Configuration

ClearView Animation Editor allows you to Copy and Paste controls animation configurations. A control configuration can be copied by using context menus. The last can be invoked by right-clicking the mouse (Figure #22.15).

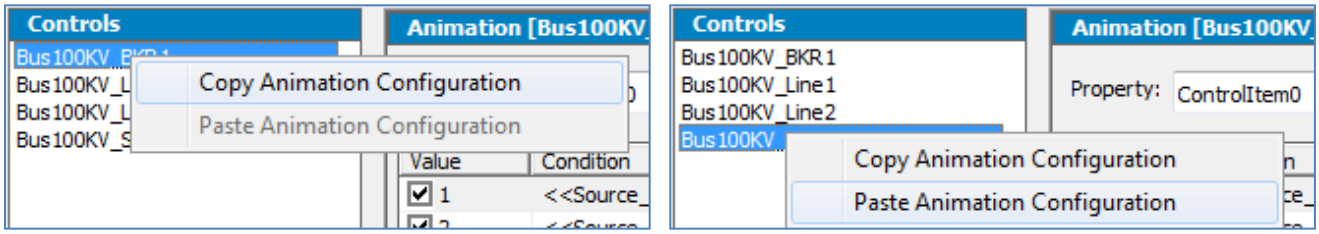


Figure #22.15

ControlItem(s) Properties States

Table #22.1

ControlItem(s) Values	Description
1	Expression evaluation to TRUE returns value of 1
2	Expression evaluation to TRUE returns value of 2
3	Expression evaluation to TRUE returns value of 3
4	Expression evaluation to TRUE returns value of 4
5	Expression evaluation to TRUE returns value of 5
6	Expression evaluation to TRUE returns value of 6
7	Expression evaluation to TRUE returns value of 7
8	Expression evaluation to TRUE returns value of 8
9	Expression evaluation to TRUE returns value of 9
10	Expression evaluation to TRUE returns value of 10
11	Expression evaluation returns Bad Tag Quality Error - tag(s) quality is BAD
Any negative value	Expression evaluation returns expression error: the error is written in the following format <Error Code>.<expression ordinal number>. See error codes below in the table for Script Error Codes Written to ControlItem(s) Properties (see below).

Script Error Codes Written to ControlItem(s) Properties

During run-time, ClearView can return next error codes in case Conditions are not as expected.

Table #22.2

Error Code	Message
11	*BAD Quality OPC Tag(s)*
-5	Invalid procedure call or argument
-6	Overflow
-7	Out of memory
-9	Subscript out of range
-10	Array fixed or temporarily locked
-11	Division by zero
-13	Type mismatch
-14	Out of string space
-28	Out of stack space
-35	Sub or Function not defined
-48	Error in loading DLL
-51	Internal error
-53	File not found
-57	Device I/O error
-58	File already exists
-61	Disk full
-67	Too many files
-70	Permission denied
-75	Path/File access error
-76	Path not found
-91	Object variable or With block variable not set
-92	For loop not initialized

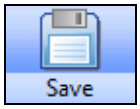
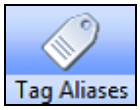
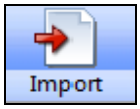
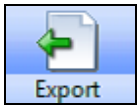
Error Code	Message
-94	Invalid use of Null
-322	Can't create necessary temporary file
-424	Object required
-429	ActiveX component can't create object
-430	Class doesn't support Automation
-432	File name or class name not found during Automation operation
-438	Object doesn't support this property or method
-440	Automation error
-445	Object doesn't support this action
-446	Object doesn't support named arguments
-447	Object doesn't support current locale setting
-448	Named argument not found
-449	Argument not optional
-450	Wrong number of arguments or invalid property assignment
-451	Object not a collection
-453	Specified DLL function not found
-455	Code resource lock error
-457	This key already associated with an element of this collection
-458	Variable uses an Automation type not supported in VBScript
-500	Variable is undefined
-501	Illegal assignment
-502	Object not safe for scripting
-503	Object not safe for initializing

Error Code	Message
-1001	Out of memory
-1002	Syntax error
-1003	Expected ':'
-1004	Expected ';'
-1005	Expected '('
-1006	Expected ')'
-1007	Expected ']'
-1008	Expected '{'
-1009	Expected '}'
-1010	Expected identifier
-1011	Expected '='
-1012	Expected 'If'
-1013	Expected 'To'
-1014	Expected 'End'
-1015	Expected 'Function'
-1016	Expected 'Sub'
-1017	Expected 'Then'
-1018	Expected 'Wend'
-1019	Expected 'Loop'
-1020	Expected 'Next'
-1021	Expected 'Case'
-1022	Expected 'Select'
-1023	Expected expression

Error Code	Message
-1024	Expected statement
-1025	Expected end of statement
-1026	Expected integer constant
-1027	Expected 'While' or 'Until'
-1028	Expected 'While', 'Until', or end of statement
-1029	Too many locals or arguments
-1030	Identifier too long

Animation Editor Toolbar

The Animation Editor window has a toolbar that provides the following functionality:

	Saves Animation Configuration.
	Edits Tags Aliases.
	Imports exported configuration.
	Exports configuration.
	Closes Animation Editor.

Configuration Export Format (.csv)

When Export is requested from the Animation Editor window, all the data is saved in a comma-separated (*.csv) file which can be reviewed with Excel.

Note: Controls that have no properties assigned are not saved in Export file.

Table #22.3

DateTime	10/22/2013 18:12										
Source Tags	Source Alias										
_SysConsumerBackupIP	Tag1										
_SysPrimaryServerRole	Tag2										
_SysSquare	Tag4										
CVSecurityLogin	Tag6										
Triggered Tags	Triggered Alias										
_SysRedundancySwitch	Tag3										
_SysTickMin	Tag5										
Object Name	ControlItem(s)	State-1	State-2	State-3	State-4	State-5	State-6	State-7	State-8	State-9	State-10
ATextCtl45	ControlItem0	<<Tag1>> = 50	<<Tag2>> = 120								
BKR6	ControlItem3					<<Tag3>> = 100					

Suspending and Resuming Animation

ClearView Animation can be suspended or resumed when necessary. When suspended, the animation script is not executed at Run time, resuming animation resumes execution of the animation script.

Suspending and resuming can be done from a context menu activated by right-clicking the active screen (Figure #12.16).

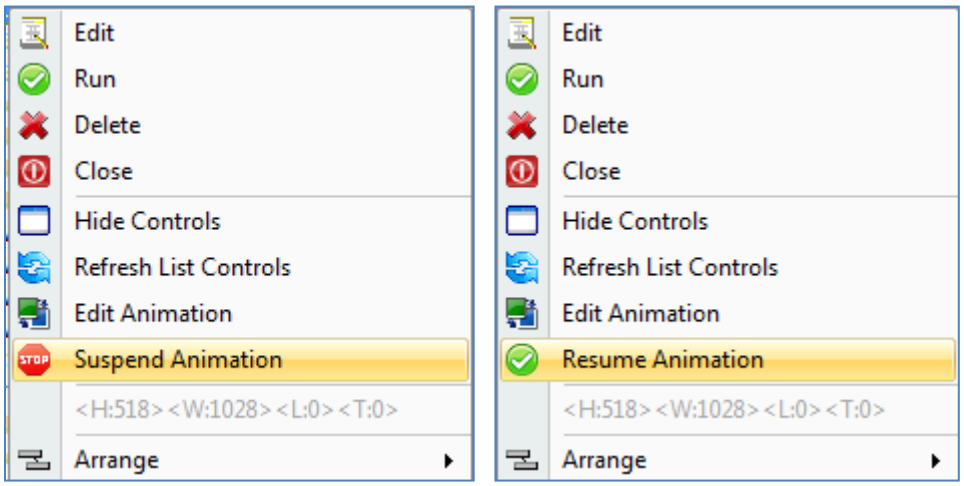


Figure #22.16

Implementation

Before you configure the animation, add a Wizard Script to the object that you want to animate. You can create your Wizard Script for one object and it you copy the object the script will be copied too. If, for example, the object name you want to animate is 35KVLine1, then the copied object would receive the new name 35KVLine2 and all associated scripts would be transposed to the new object.

Example - 35KVLine1 Object

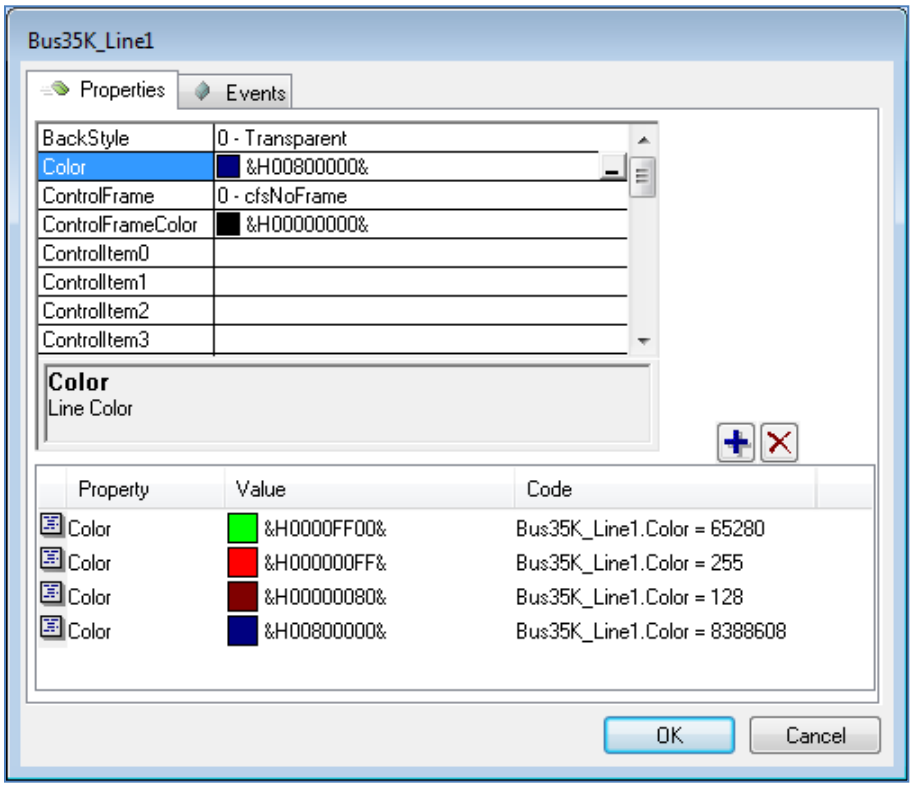


Figure #22.17

Table #22.4

Object Color	Expression
Green	Bus35K_Line1.ControlItem0 = 1
Red	Bus35K_Line1.ControlItem0 = 2
Brown	Bus35K_Line1.ControlItem0 = 3
Blue	Bus35K_Line1.ControlItem0 = 4

A previously configured animation will automatically set ControlItem(s) property according to configured expressions.

SECTION 23

Screen zooming and resizing

ClearView allows you to specify how object will be located on the screen at runtime. The following modes are available: Original size, Fit Screen, Fit Screen Width, Fit Screen Height, and Zoom. The Zoom mode allows you to change the zoom ratio at runtime, either programmatically or with the keyboard or mouse.

Zoom and Resize properties can be changed either in Design mode by changing Drawing Pad properties or at runtime from the script.

Setting the Drawing Pad Resize Properties

Switch the Drawing Pad to Design mode. The Pad properties will appear as shown in Figure #23.1.

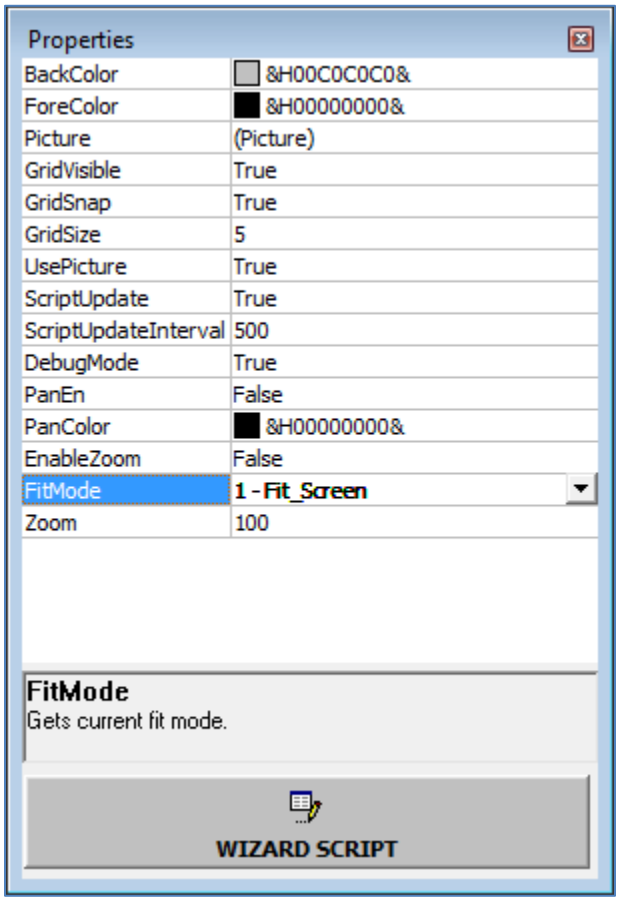


Figure #23.1

Property	Description	Parameters
FitMode	Selects type of fit mode which would be used	0-Original_Size – set the screen to original size 1-Fit_Screen – Fit screen width and height 2-Fit_Width – Fit to screen width 3-Fit_Height – Fit to screen height 4-Zoom – User defined screen size (zoom/un-zoom)
Zoom	Percent zoom (Integer). Works in conjunction with FitMode property set to “Zoom”	

Accessing Screen Resizing Property via Scripting

Code	Description
Pad.FitMode = 0	Set ClearView-SCADA screen to original size
Pad.FitMode = 1	Set ClearView-SCADA screen to fit the monitor (width and height)
Pad.FitMode = 2	Set ClearView-SCADA screen to fit the monitor width
Pad.FitMode = 3	Set ClearView-SCADA screen to fit the monitor height
Pad.FitMode = 4	Proportional user configurable screen resize
Pad.Zoom = 50	Screen size is set to 50%

Run-time Mouse and Keyboard Zoom Functionality

If FitMode is set to Zoom:

- Getting focus to the Drawing Pad and holding the Control (Ctrl) key and plus (+) key would increase the size (zoom-in)
- Getting focus to the Drawing Pad and holding the Control (Ctrl) key and minus (-) key would decrease the size (zoom-out)
- Getting focus to the Drawing Pad and using mouse scroll forward would increase the size (zoom-in)
- Getting focus to the Drawing Pad and using mouse scroll backward would decrease the size (zoom-out)

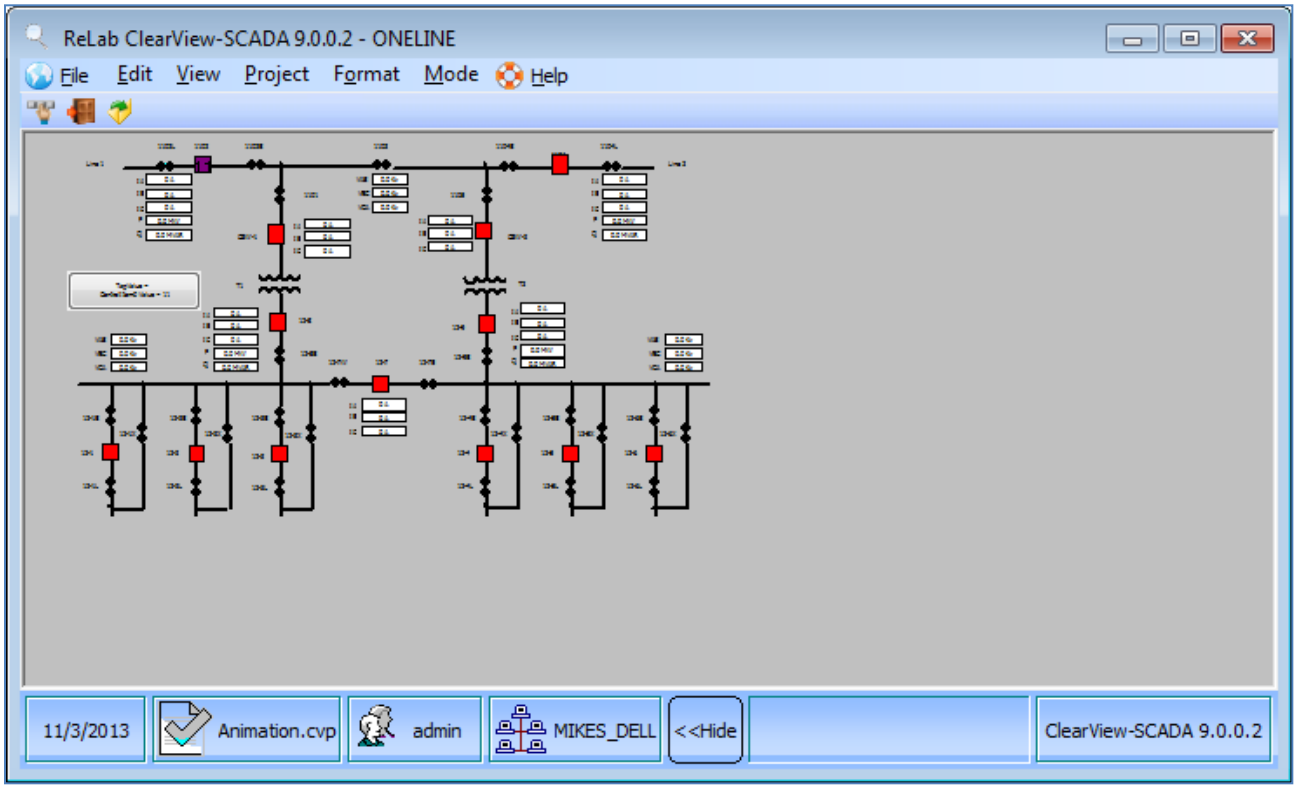


Figure #23.2

SECTION 24

Lockout/TagOut

Overview

North American Electric Reliability Corporation (NERC) and Critical Infrastructure Protection (CIP) Common standards require that any mechanical or electrical equipment can be locked out and tagged out before being worked on.

ClearView-SCADA lock-out/tag-out function ensures that controllable objects (tags) in the application are properly secured prior to and during maintenance or servicing work. The tag-out state can easily be identified by the user. When a specific Data Item is tagged a user would not able to execute a write to the specified lockout tag.

In multiuser environment number of people can lockout specific equipment. All users are allowed to request a supervisor acknowledgement in order to remove the equipment lockout. The supervisor acknowledgement can be done only after all users would remove equipment lock.

ClearView-SCADA provides a historical view of all changes which were made to the system. The changes such as lockouts can be easily retrieved and analyzed.

ClearView-SCADA Client LockOut/TagOut Interface

Every time a user clicks to open LockOut/TagOut Interface the user must provide login information (User Name and Password)

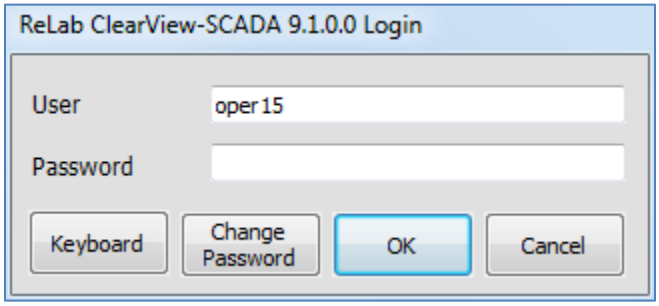


Figure #24.1

Upon successful login the LockOut/TagOut screen is opened in Current Tagging mode.

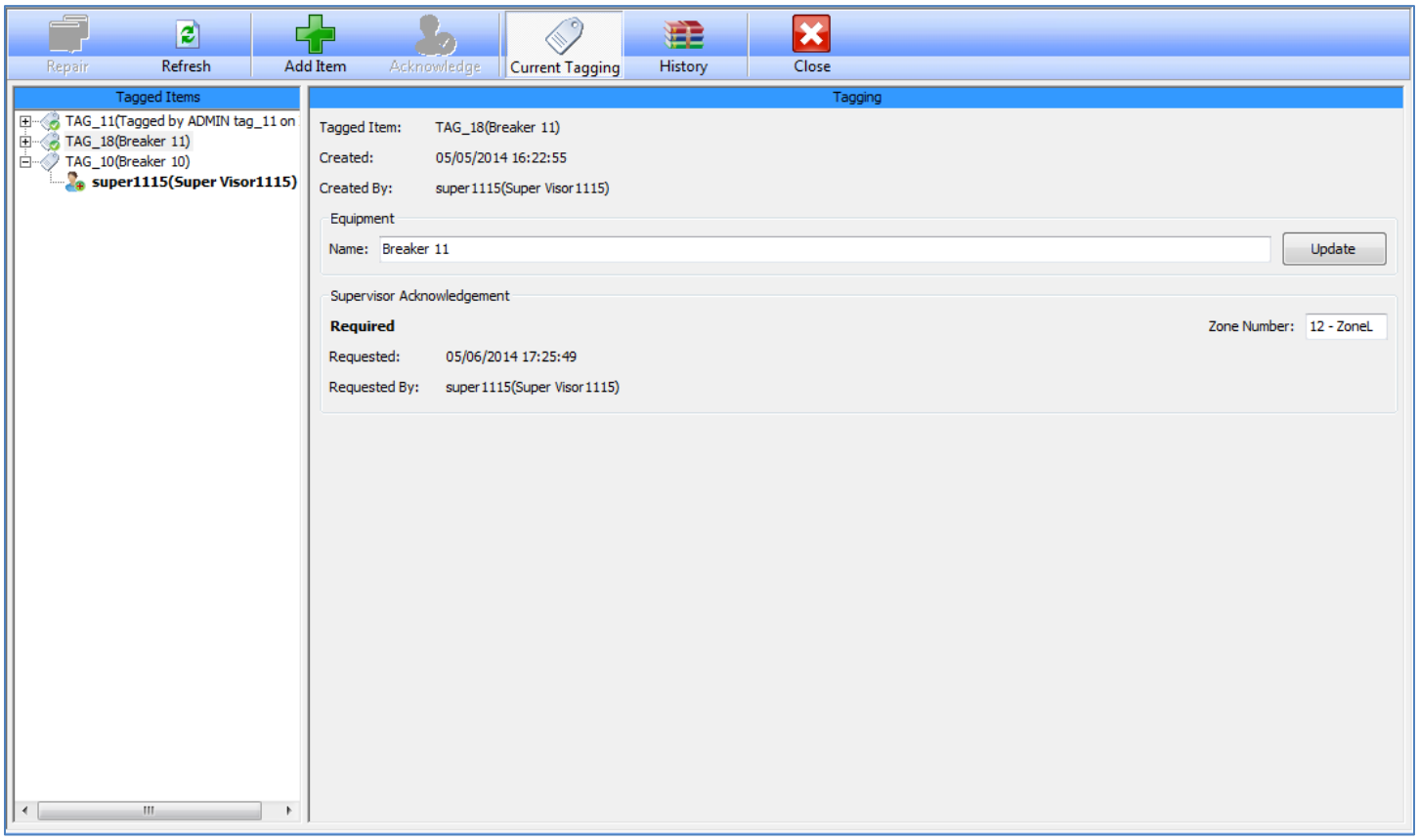


Figure #24.2

TagOut/LockOut screen toolbar

Repair - becomes activated if error writing to database is detected, so it must be repaired by the same user that has set Tagging for Data Item.

Refresh - will refresh the list of Tagged Items (in case several Clear View instances are working in parallel, so information will be synchronized for all users on Refresh) either in Current Tagging or in History view.

Add Item – starts Add Item Wizard to choose and to assign tagging for Data Item.

Acknowledge – activated only when all users had removed tagging from a particular Data Item and current user has permissions as Supervisor according to zone number.

Current Tagging – switching from History view to Current Tagging view.

History – switching from Current Tagging view to History view.

Close - exit from Tagging window

Current Tagging mode has two main windows: **Tagged Items** and **Tagging**

Tagged Items list displays Data Items with all users (displaying User Name, First Name, and Last Name) that had already set tagging. Changing focus on the lines in Tagged Items list will show all information available for tagged items on the right side in Tagging window. A user will have all permissions to access “his” tagged items (tagged items set by this user), the ability to update/clear the existing tagging, add tagging to already tagged items or create tagging for new items. A user can also view another users’ tagging.

Adding new Data Item

1. Click Add Item.

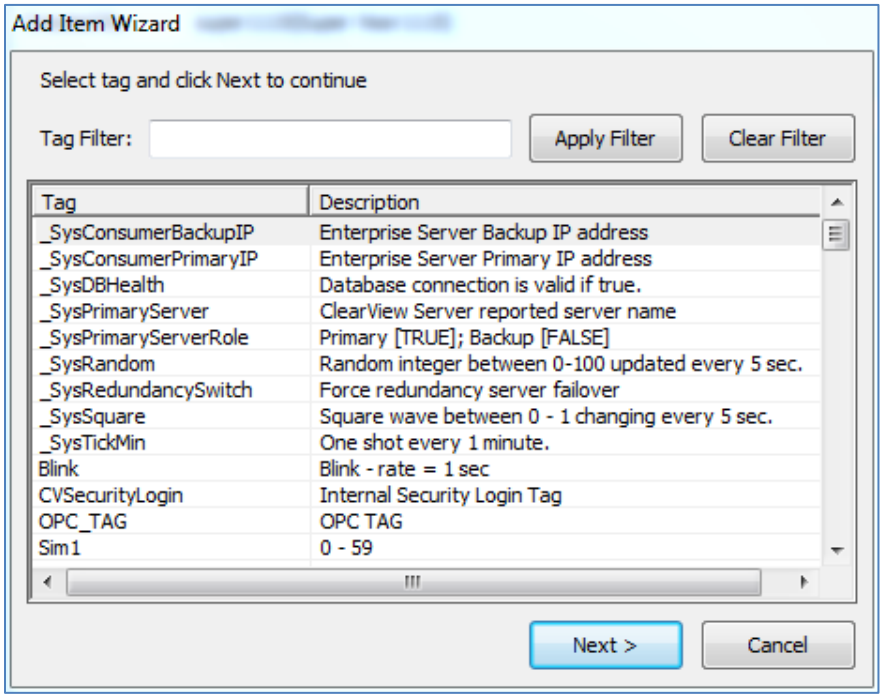


Figure #24.3

2. Apply filter to find required Tag from Tag Database in ClearView.
3. Select tag (Data Item) and click next

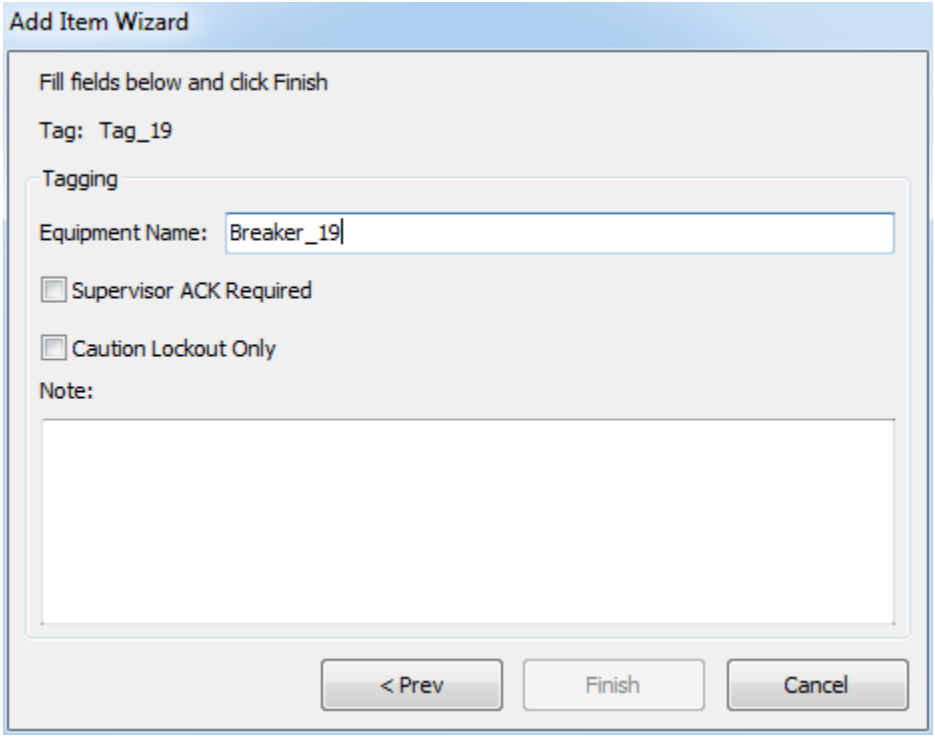


Figure #24.4

4. Provide mandatory equipment name and Note.
5. Press Finish.
6. Optionally you can add “Supervisor ACK required” of tagging removal. In this case you must provide zone number for supervisor.
7. Checking “Caution Lockout Only” will not lock the tag but will mark the tag with a “Caution” flag. The last means that if somebody will try to change the tag then the “Tag lockout warning” (Figure 24.5) message box will pop up. Clicking “Yes” will write a value to the tag, Clicking “No” will cancel the operation.

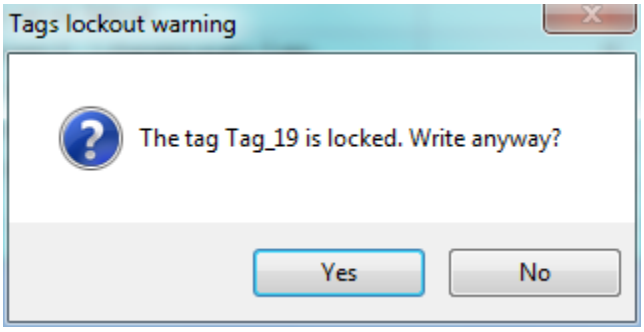


Figure #24.5

Note: Any user with permissions for the zone is considered as a Supervisor for the Tagging meaning he can acknowledge and remove tagging for a Data Item.

Adding a Tag to already tagged Item:

1. Select the item in Tagged Items window.
2. Add comments in Current User area and click Create Tagging. If Supervisor acknowledge had not been set yet, it can be set now.

Removing (clearing) locked out equipment.

When a particular user is ready to clear tagging from an item the user must provide the comments and press clear tagging.

Supervisor Acknowledgement

If current user has the same zone permissions as it is set for the acknowledgement, the Acknowledge button is enabled. Pressing Acknowledge will clear the tagging and the Data Item will be moved from Current Tagging to History.

Tagging History

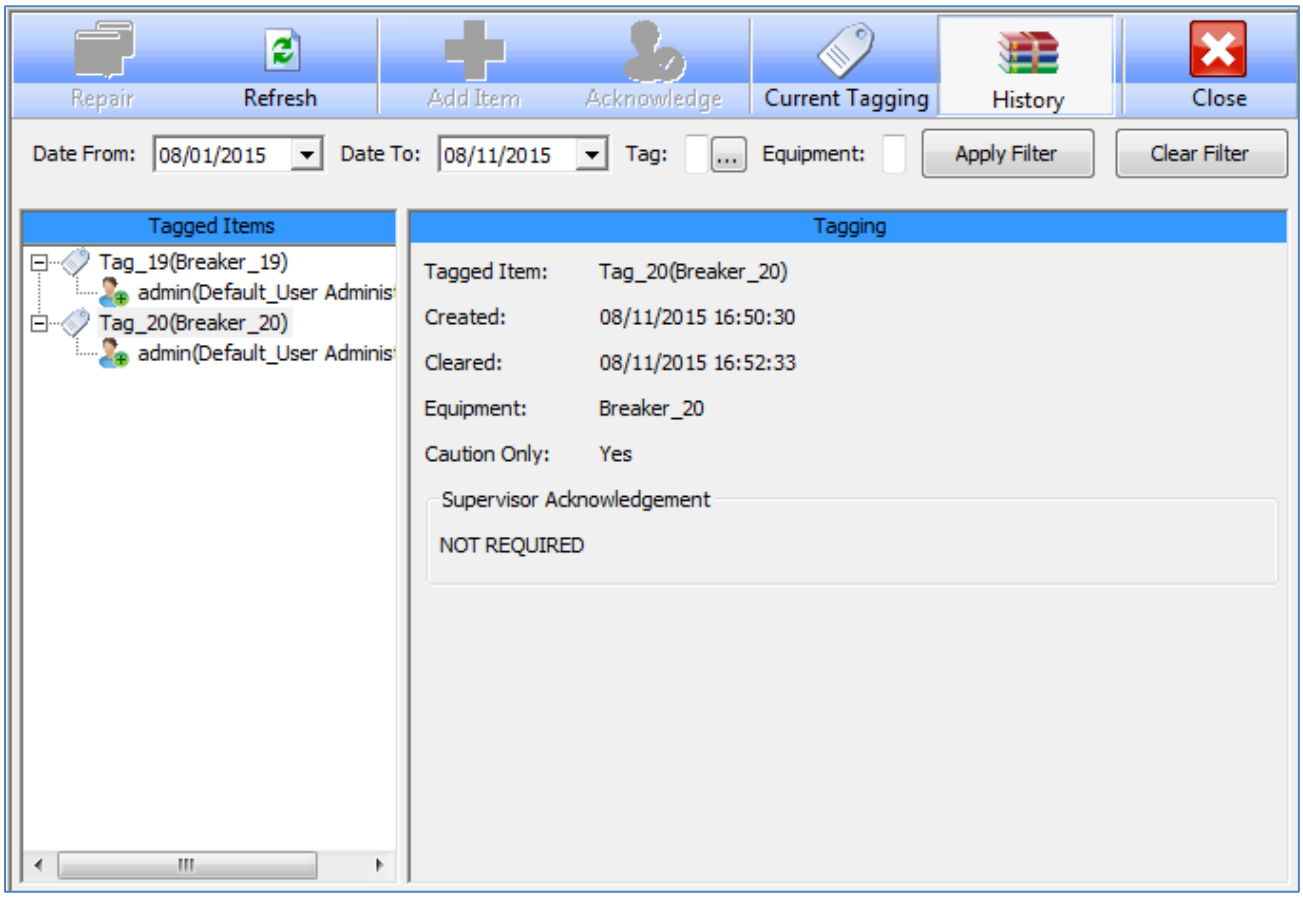


Figure #24.6

History view has two windows similar to Current Tagging – Tagged Items and Tagging and tagging information for all cleared Data Items. The view can be filtered by date (from/to), tag name and equipment name.

ClearView Database Redundancy and Tagging

Tagging information is stored in ClearView database.

If redundant ClearView server and redundant database are not specified then on tagging changes ClearView client updates database and then sends update notification to ClearView server. If at the moment of changes ClearView Server is not running the changes are still saved in data base and server will read tagging information upon startup.

If redundant ClearView server and redundant database are specified in the configuration the client will update primary database and then update backup database. If for some reason backup database is not accessible the ClearView Client will revert the changes in primary database. ODBC error message will be displayed. In the last case the changes to tagging will not be made. To resolve this situation a user needs to make sure that both (primary and redundant) databases are available at the moment of changes.

In the case if primary or redundant ClearView Server is not available the Tagging Editor will generate the following message (Figure 24.7).

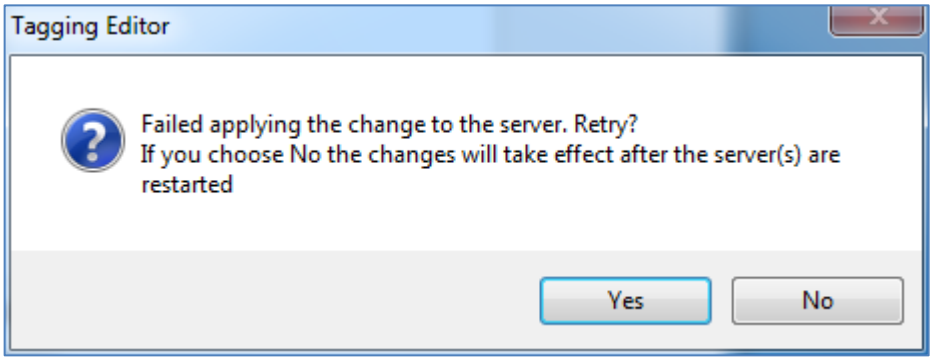


Figure #24.7

- Pressing “Yes” will repeat tagging request;
- Pressing “No” will keep tagging information in database but will not propagate it to ClearView servers. The record will be displayed with an exclamation mark (Figure 24.8).

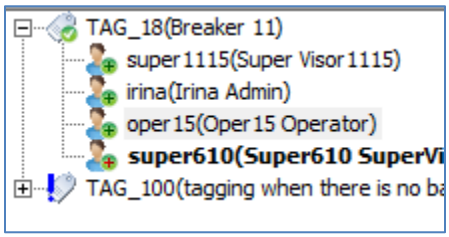


Figure #24.8

The exclamation mark means that such record will become active only after Repair is pressed. Option to Repair the records is available when both CV Servers are running and accessible.

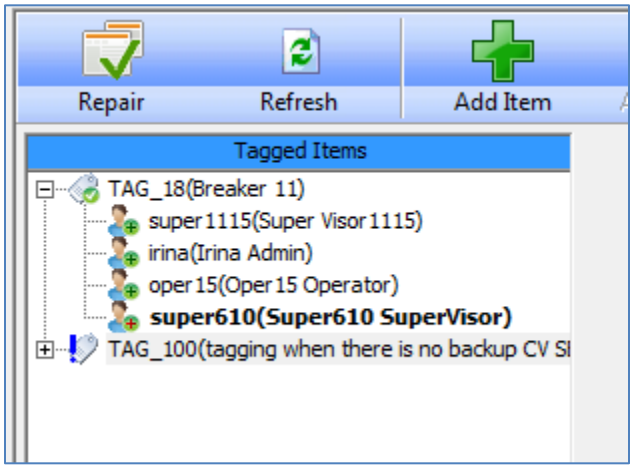


Figure #24.9

After successful Repair the record becomes active meaning the Data Item is tagged.

Note that Repair action for a record can be performed only by the user who created the record.

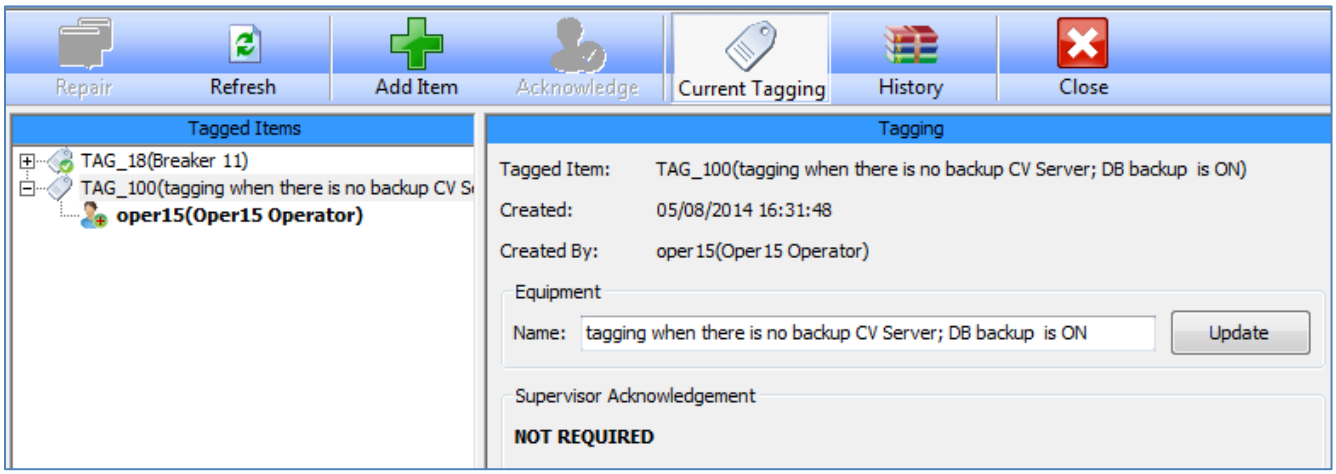


Figure #24.10

Using Lockout information in ClearView scripts

User can access Lockout information from within ClearView scripting or from ClearView Wizard. User can use Tag Object's *Tagged* property for this purpose.

If *Tagged* property is set to *true* – the data point (equipment) associated with this tag is locked out.

If *Tagged* property is set to *false* – the data point (equipment) associated with this tag is not locked out.

Scripting example:

```
If  Tags.item("BREAKER 1").Tagged = True   then
        'Indicate that the BREAKER 1 is locked out here
Else
        'Indicate that the BREAKER 1 not locked out here
End If
```

SECTION 25

Users database synchronization module (CVDBSync)

ReLab ClearView SCADA includes a module that performs Users and User Groups synchronization between multiple instances of ClearView Database.

CVDBSync files

1. CreateMR.sql – script to create Master Repository if MSSQL is used;
2. CVDBSync.cfg.xml –saved configuration;
3. CVDBSyncConsole.exe – configuration utility to create configuration, register, start and stop CVDBSyncSvc.exe service;
4. CVDBSyncExe.exe – internal use;
5. CVDBSyncLib.dll – internal use;
6. CVDBSyncSvc.exe – service for User(s) database synchronization, appearing under the name CVDBSyncSvc in Services when registered;
7. DBSyncLR.mdb – internal application use, Local Repository for application, must not to be moved or changed;
8. DBSyncMR.mdb – Master Repository for MS Access type of DB. Must be the only one for the whole set of Clear View SCADA User databases assigned for synchronization;

Where to install

- One CVDBSync Service is required per one instance of Clear View SCADA database, so you need to install as many instances of CVDBSync Services as the number of CV databases you need to synchronize.
- Only one CVDBSync Service per host can be installed.
- If you are using MS Access for Clear View SCADA databases it is advised to install CVDBSync Service on the same hosts where the Clear View databases reside. Otherwise it is advised to setup CVDBSync Service on the same host as Clear View SCADA that has access configured to Clear View database.
- Master Repository must be the only one for all CVDBSync Services, and must be accessible for read/write operations from all hosts where CVDBSync Services installed.

Installation procedure

1. Install CVDBSync Service on all hosts included into synchronization process and install only one Master Repository for the whole system.
2. Follow the installation procedure and choose what you need for each host. During the installation you will be asked what type of setup you want, see Figure #25.1.

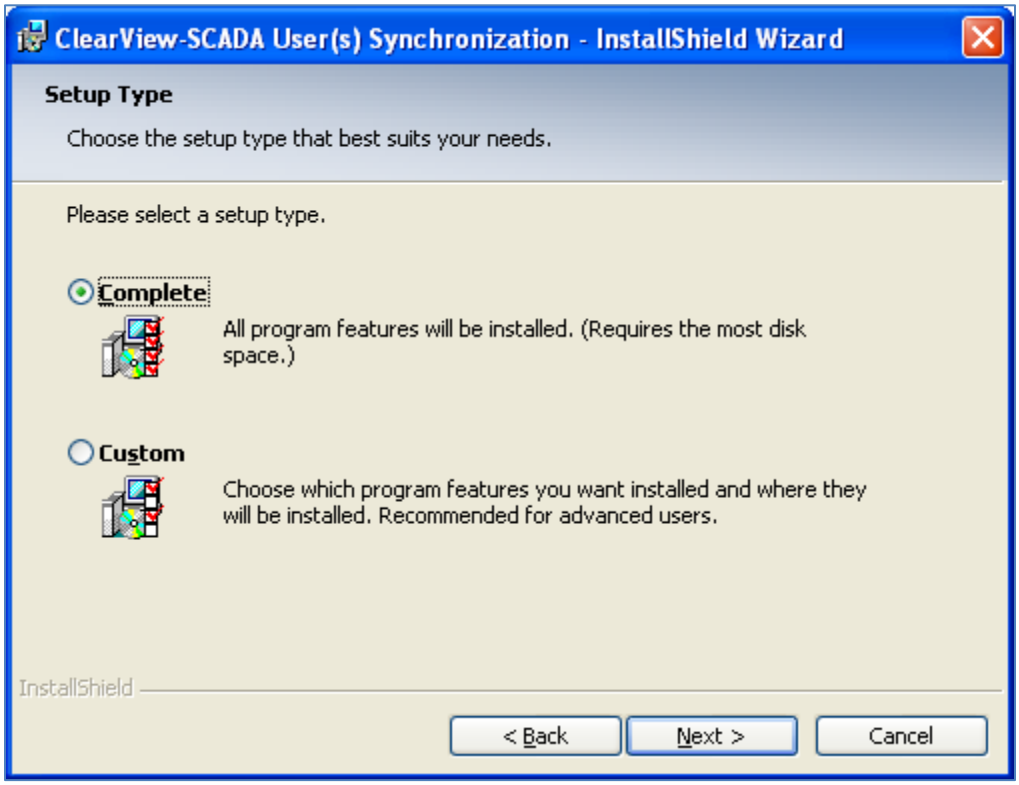


Figure #25.1

If you choose **Complete** - the Master Repository will be installed to specified location.

Custom setup type allows installing either CVDBSync Service with Local Repository or only Master Repository (no service files will be included) or both (default).

Custom screen is shown on Figure #25.2.

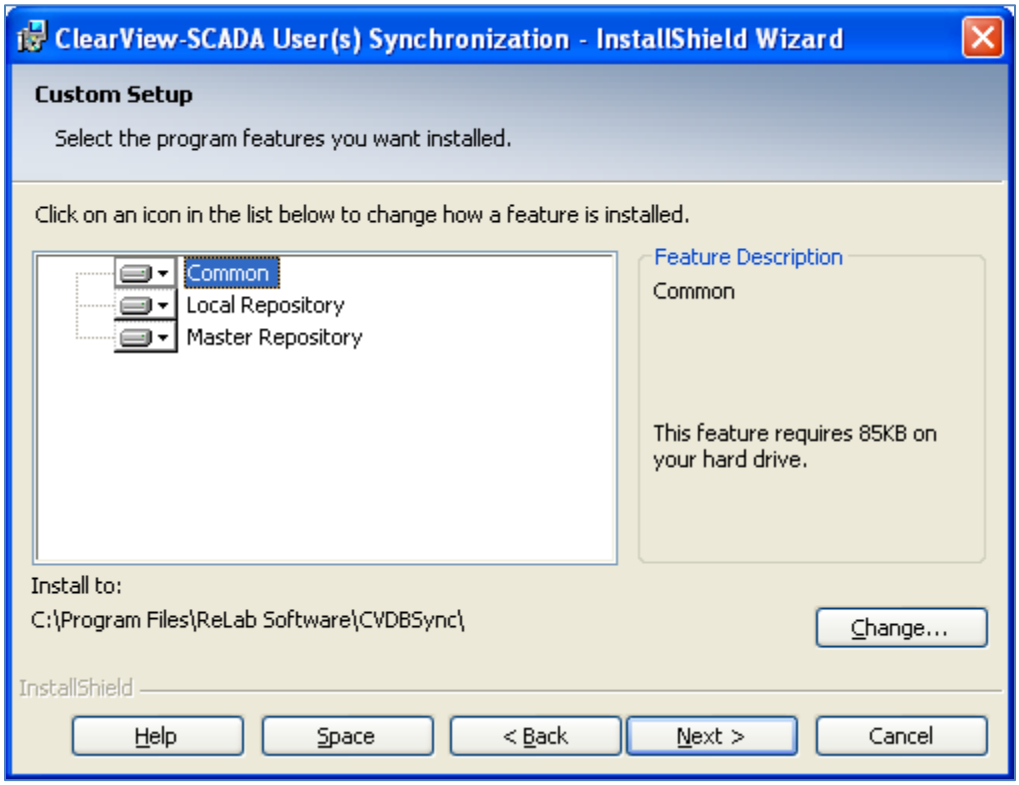


Figure #25.2

- Choose the features you need, change installation directory if needed and press Next.
- Follow the installation wizard to finish the installation.

Note: Master Repository files will be ignored if not used (not configured to use).

Warning: When you install CVDBSynch Service an empty file for Local Repository (DBSyncLR.mdb) is always present. Do not change or move it.

Configuration

Make sure to provide full permissions to access Master Repository. If you use MS Access for Master Repository – it will be the directory where the file DBSyncMR.mdb resides.

If you use MS SQL for Master Repository - create a database in MSSQL Server and run script CreateMR.sql to create required database structure.

Create DSN to access Master Repository. It must be done on each host where CVDBSync Service is installed.

1. Go to Control Panel;
2. Open Administrative Tools;
3. Open Data Sources (ODBC);
4. Open tab System DSN and add either MSSQL type of DSN or MS Access type in case you choose to use file DBSyncLR.mdb as Master Repository.
5. Create DSN for CVDBSync Service to access Clear View SCADA database (Figure 3).

6. Start CVDBSyncConsole.exe, open tab Configuration (Figure 3) and provide DSN name to access Master Repository and DSN to access Clear View Users database. Configure also how often service should access Clear View database to check for an update and how often service should report to Master Repository about the changes in local Clear View Users database.
7. Provide user names and passwords For MSSQL type of Master Repository and Clear View database.
8. Save configuration.

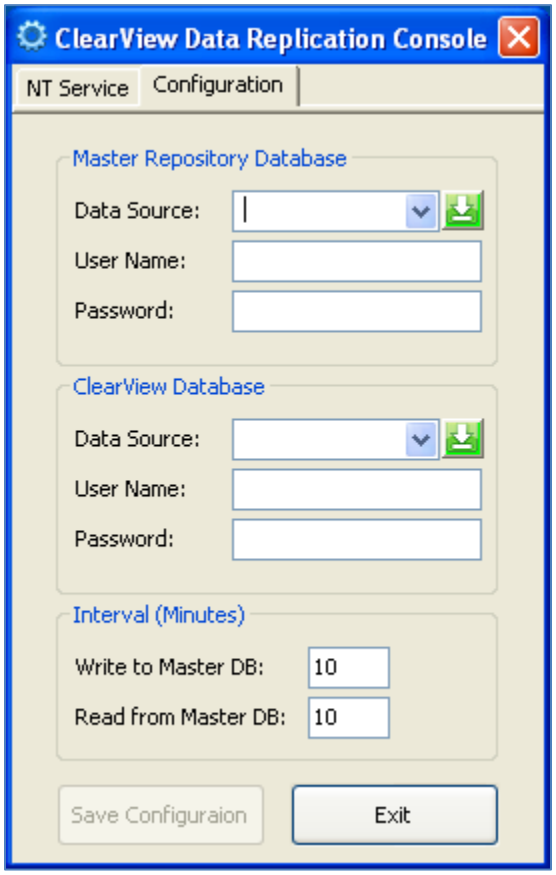


Figure #25.3

- Switch to NT Service tab.

Registration of CVDBSync Service

In order to start CVDBSync Service as service it is not sufficient for a user who is performing installation and configuration to be a local administrator. Special permissions “Log On as a Service” must be granted before service is started.

To grant a user logon service access:

1. Click Start > Settings > Control Panel.
2. Double-click Administrative Tools.
3. Double-click Local Security Policy.
4. Open Local Policies.

5. Open User Rights Assignment.
6. Open Log On as a Service.
7. Click Add.
8. Select the user you want to grant logon service access to and click OK.
9. Click OK to save the updated policy.

Provide information for the account that should be used on a host to run a service (Figure 4) and register it by clicking on “Register” icon.

Local System account can be used only on the host where Master Repository is located together with Local Repository.

Otherwise provide the name and password for a user who has permissions to start the service. The same user must have permissions to read and write into the location with Master Repository.

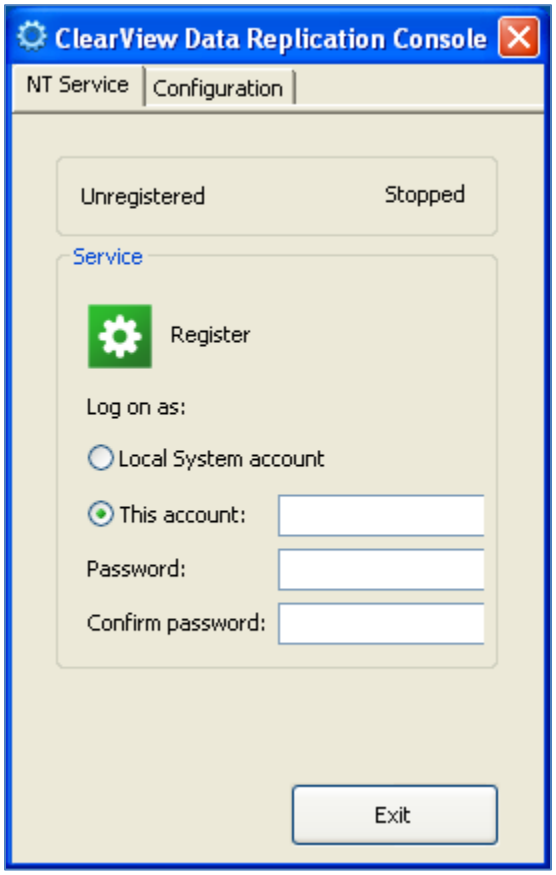


Figure #25.4

Starting / stopping CVDBSync Service.

To make sure that service starts with system boot it is recommended to use the Automatic type of service (Figure #25.5).

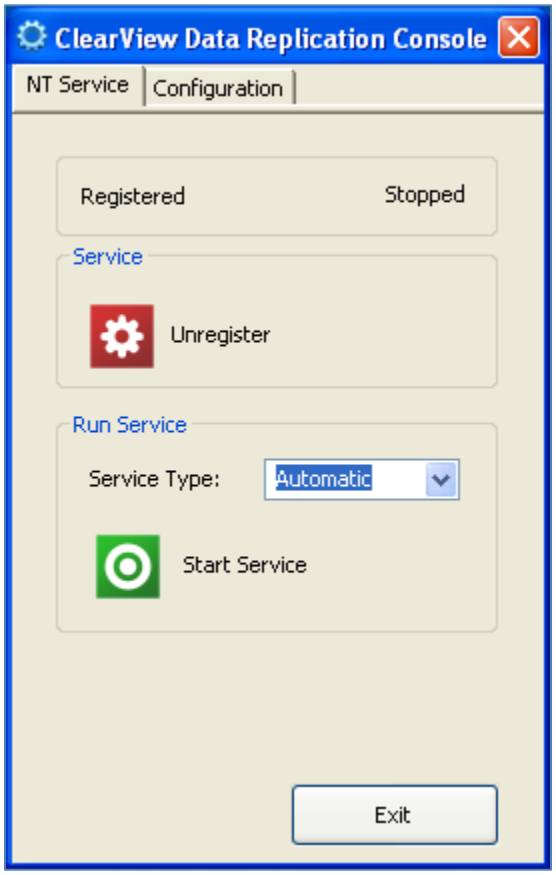


Figure #25.5

To stop CVDBSync Service start CVDBSyncConsole.exe and click “Stop Service” (Figure 6) or use Services from Control Panel.

When service is stopped you may unregister it (Figure #25.5) by clicking “Unregister”.

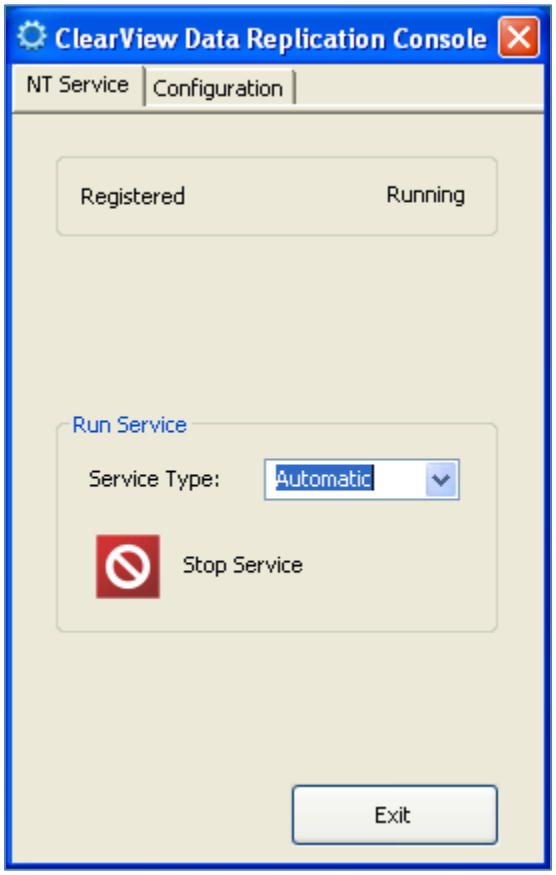


Figure #25.3

Note: It might happen that during the time service was requested to be terminated it was at the stage of accessing database. In this case it will continue to run in order to avoid database corruption. In this case (even status for service is not defined anymore as running) you will see CVDBSyncSvc.exe on the list of running processes (make sure you check "Show processes from all users" to see it). The safest way to start service again is to reboot a host (if service is assigned to be run automatically). For manual type of service reboot the host and start service again manually.

The principles of synchronization

- When each CVDBSync Service is started it is checking the content of Master Repository and comparing it to Local Repository content (the last one is always reflecting the content of Clear View SCADA Users database assigned to this particular service).
- All users and groups missing in local Clear View database will be added to the ClearView database. All users and groups missing in Master Repository will be added to Master Repository, so another service will read it from Master Repository and add into another Clear View database.
- Users are identified by User Name. For the initial time of synchronization an update for the same user will be done in order of starting CVDBSync Services.
- In rare occasions it might happen that a user A was removed from already synchronized Clear View database (CV DB 1) but the same user A still present in not yet synchronized Clear View database (CV DB 2). Then at the moment of including CV DB 2 into the system user A will be added again to Master Repository and replicated to all Clear View databases assigned for synchronization. In this case such user should be deleted again manually and then it will be deleted from all Clear View databases included into synchronization.

- Operations with User or Group record in Clear View database (deleted, updated or added) will be reported into Master Repository by local service and replicated to all hosts by other services.

Troubleshooting

- If replication services experience a problem to access Master Repository because of the database corruption it might be restored to the initial state without stopping local services. It is recommended to keep copy of an empty Master Repository file or MSSQL script for creating Master Repository from scratch in case something goes wrong with Master Repository database. Make sure host with Master Repository location is NOT accessible for the local services during fixing a problem.
- Replacing Master Repository with an empty one or restoring MSSQL type of Master repository to the initial condition will fix Master Repository database issues and local services will restore the content again when Master Repository is fixed and.
- Check log files CVDBSync.log and CVDBSync_Critical.log for errors and problems. The synchronization services are able to restore connection to Clear View database and to Master Repository in cases like network disruption, SQL server unavailability, host reboot etc.

SECTION 26

ClearView Localization

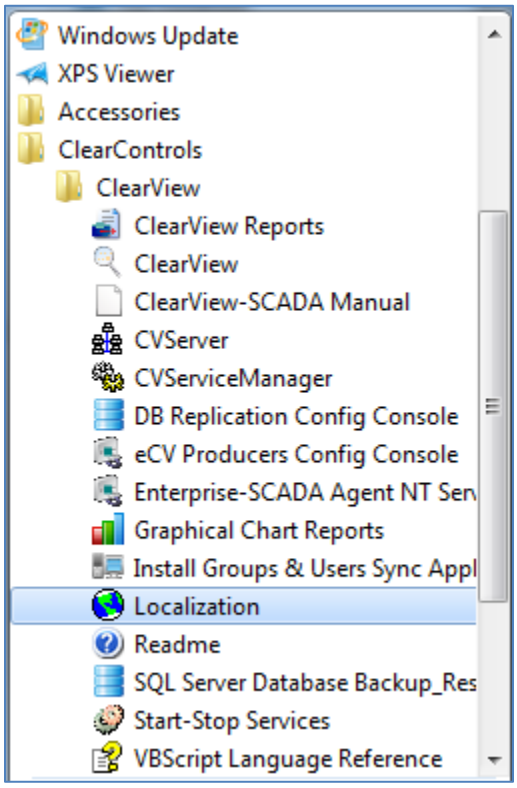


Figure #26.1

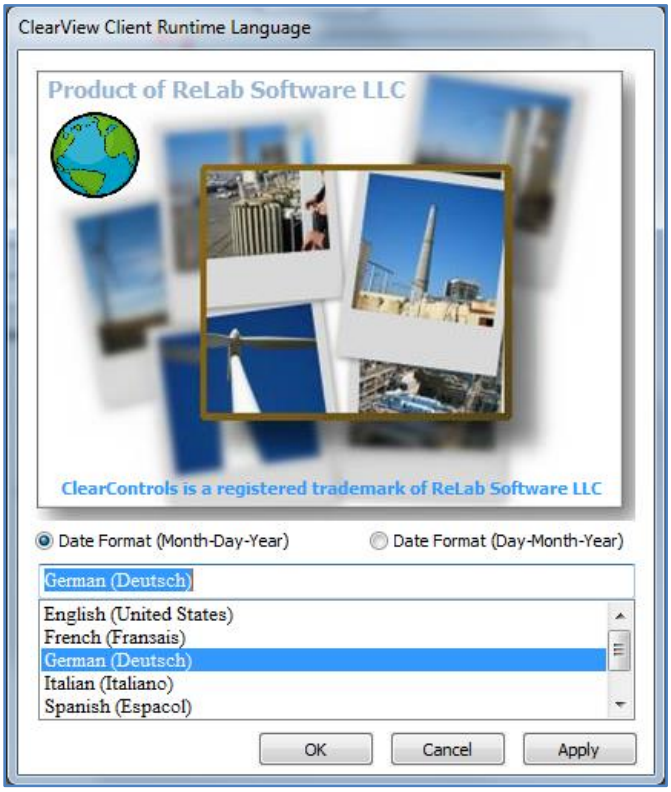


Figure #26.2

Select required language and click OK.

The regional setting interface will appear

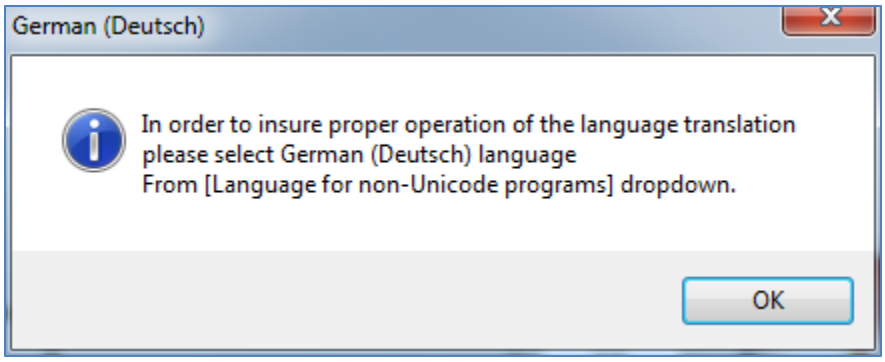


Figure #26.3

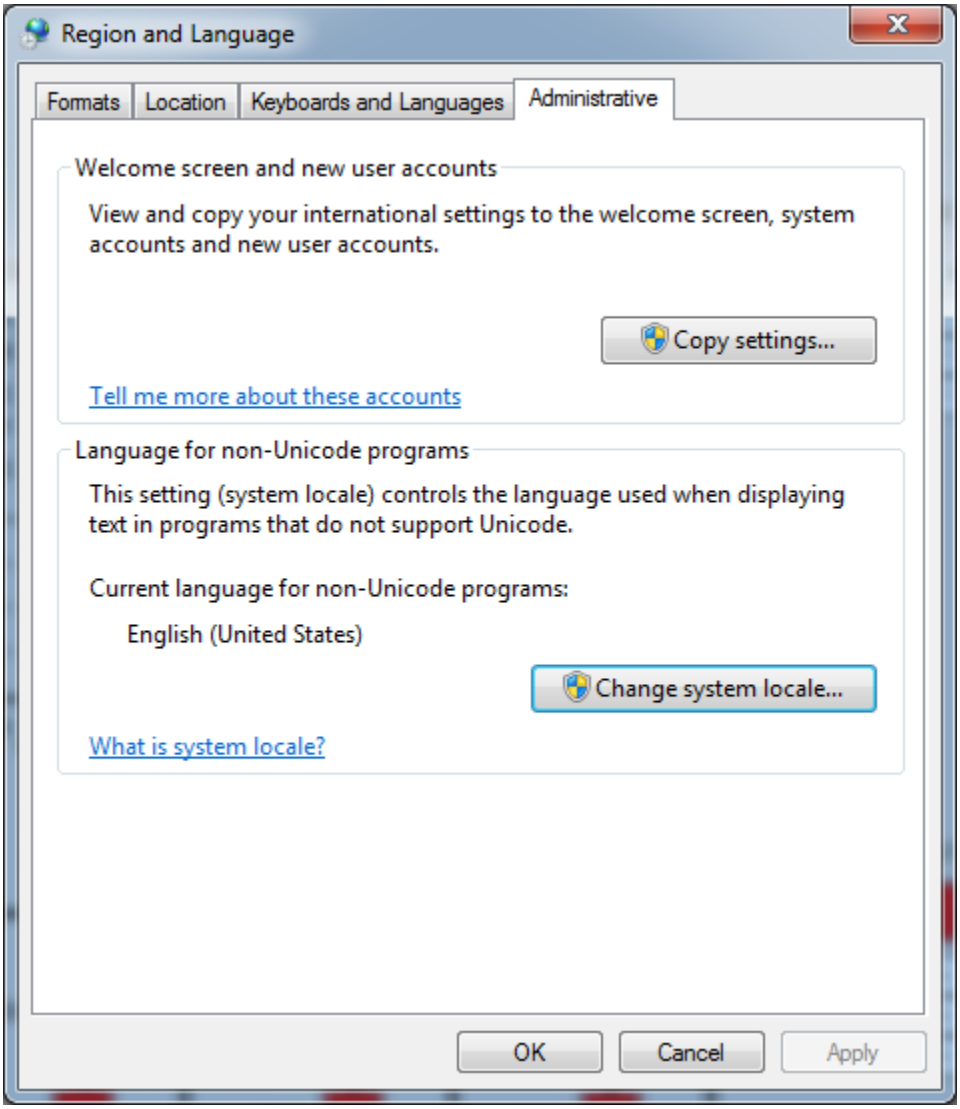


Figure #26.4

Press “Change system locale”

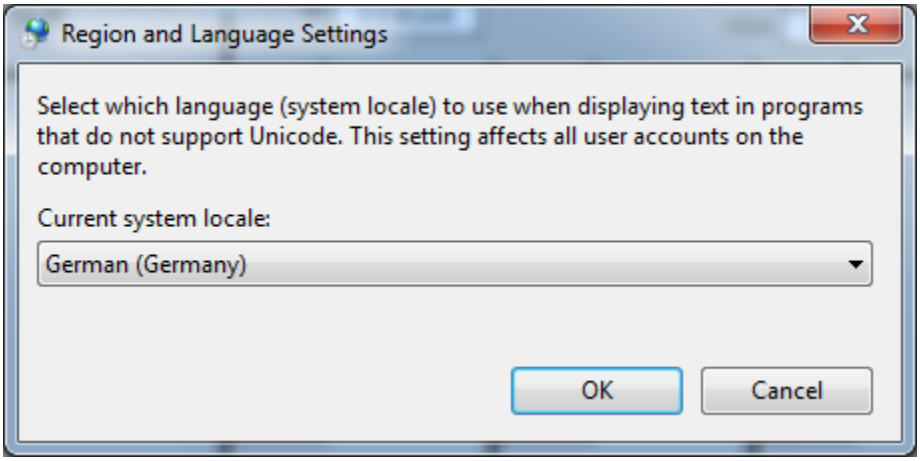


Figure #26.5

Select appropriate language from “Current System locale” dropdown and click “OK”. The system will prompt you to restart your machine.

Note: ClearView provides an ability to add additional languages programmatically. All translated words are entered into comma separated value (CSV) file. To modify or to add additional languages open and edit the CVLocalization file **Program Files\ClearControls\ClearView\Bin\Objects\CVLocalization.csv.**

SECTION 27

Text to Speech Interface

ClearView-SCADA provides an interface to Microsoft Text to Speech Engine. This functionality can be used, for example, to enable audible alarms.

Prerequisites: Microsoft Text to Speech SDK version 5.1 or higher must be installed on the computer:

- SpeechSDK51.exe - required
- SpeechSDK51LangPack.exe – Optional (if another than English Language is required)

Speech SDK 5.1 is available from Microsoft and can be downloaded from: *www.microsoft.com/en-us/download/details.aspx?id=10121*

ClearView-SCADA Scripting Example:

The following example demonstrates how an alarm description can be pronounced when the alarm is activated.

```
Dim AlarmToSpeech
Set AlarmToSpeech = CreateObject ("SAPI.SpVoice")

Private Sub SubscribedAlarms_AlarmStatusChanged (AlarmName, CurrentState)
  If CurrentState = 1 Then
     AlarmToSpeech.Speak (SubscribedAlarms.item(AlarmName). AlarmConfig.AlarmDescription & " is Active")
  End If

End Sub
```

Text to Speech Properties:

The following Text to Speech Properties are available within scripting.

Property	Description
AlertBoundary	Gets and sets the alert boundary, which specifies how a speaking voice pauses itself for alerts.
AllowAudioOutputFormatChangesOnNextSet	Gets and sets the flag that specifies whether the voice is allowed to adjust its audio output format automatically.
AudioOutput	Gets and sets the current audio output object used by the voice.
AudioOutputStream	Gets and sets the current audio stream object used by the voice.
EventInterests	Gets and sets the types of events received by the voice.
Priority	Gets and sets the priority level of the voice.
Rate	Gets and sets the speaking rate of the voice.

Status	Returns the current speaking and event status of the voice in an ISpeechVoiceStatus object.
SynchronousSpeakTimeout	Gets and sets the interval, in milliseconds, after which the voice's synchronous Speak and SpeakStream calls will time out when its output device is unavailable.
Voice	Gets and sets the currently active member of the Voices collection.
Volume	Gets and sets the base volume (loudness) level of the voice.

Text to Speech Methods:

The following Text to Speech Methods are available within scripting.

Method	Description
DisplayUI	Initiates the display of the specified UI.
GetAudioOutputs	Returns a selection of available audio output tokens.
GetVoices	Returns a selection of voices available to the voice.
IsUISupported	Determines if the specified UI is supported.
Pause	Pauses the voice at the nearest alert boundary and closes the output device, allowing it to be used by other voices.
Resume	Causes the voice to resume speaking when paused.
Skip	Causes the voice to skip forward or backward by the specified number of items within the current input text stream.
Speak	Initiates the speaking of a text string, text file or wave file by the voice.
SpeakCompleteEvent	Gets an event handle from the voice that will be signaled when the voice finishes speaking.
SpeakStream	Initiates the speaking of a text stream or sound file by the voice.
WaitUntilDone	Blocks the caller until either the voice has finished speaking or the specified time interval has elapsed.

For more information see Automation SPVoice visit: *https://msdn.microsoft.com/en-us/library/ms723602%28v=vs.85%29.aspx*

APPENDIX A

OLE for Process Control Overview (OPC)

This specification describes the OPC COM Objects and their interfaces implemented by OPC Servers. An OPC Client can connect to OPC Servers provided by one or more vendors.

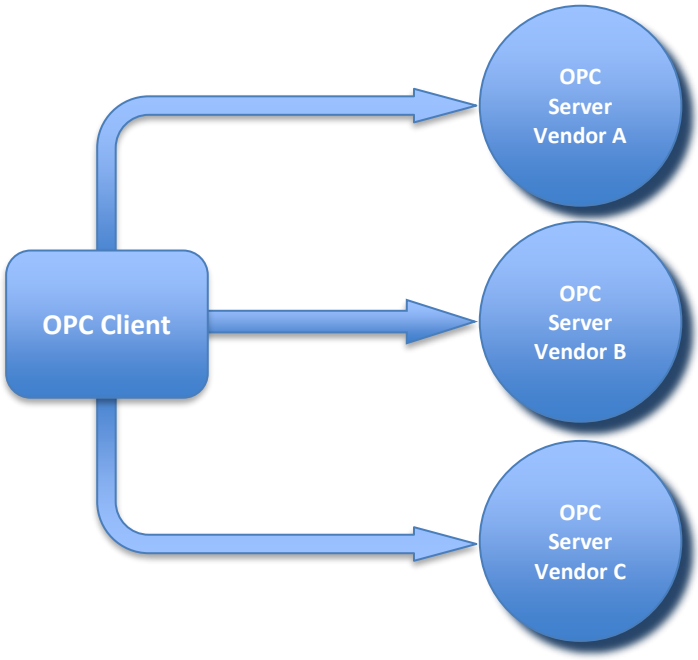


Figure #A.1 - OPC Client

Different vendors may provide OPC Servers. Vendor-supplied code determines the devices and data to which each server has access, the data names, and the details about how the server physically accesses that data. Specifics on naming conventions are supplied in a subsequent section.

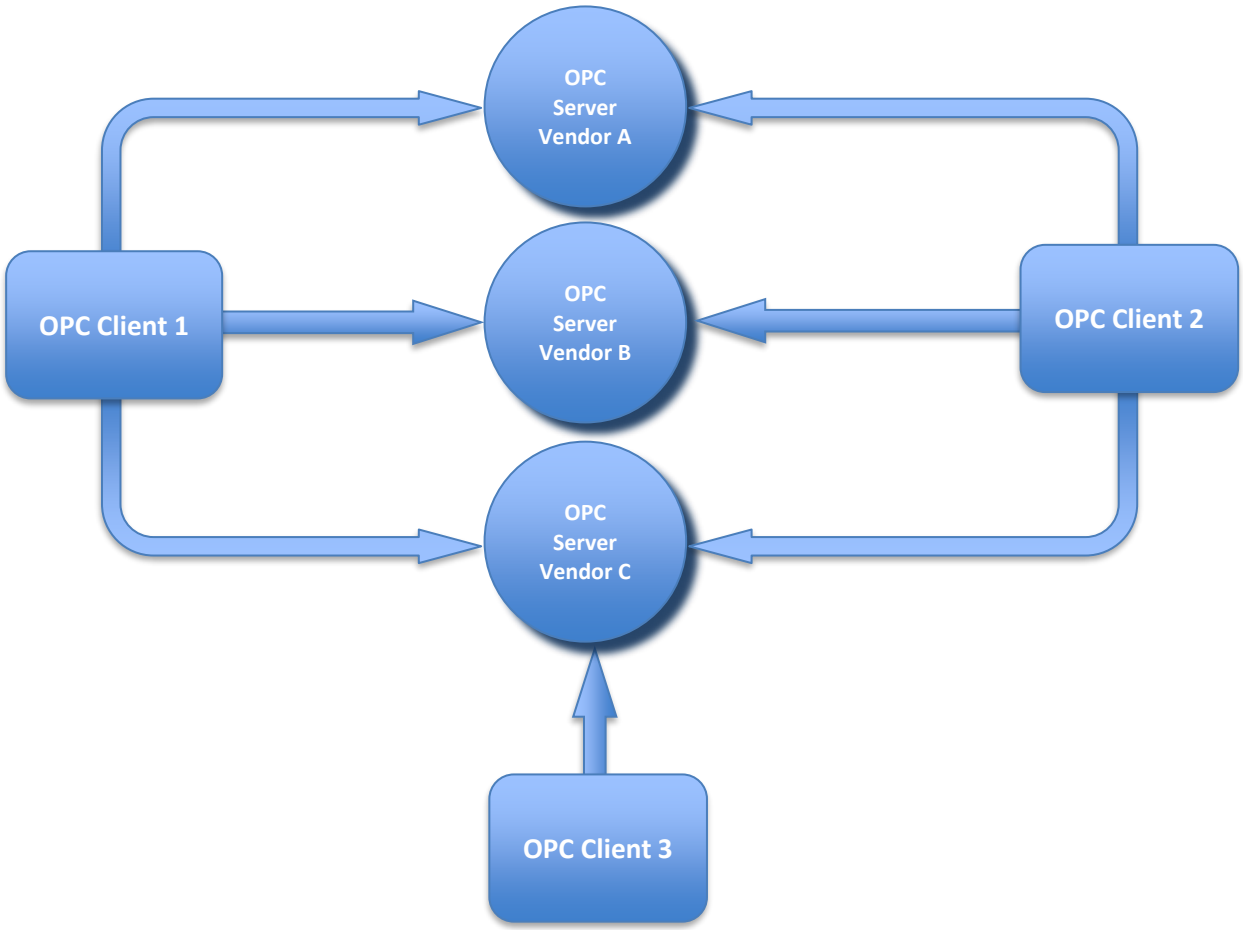


Figure #A.2 - OPC Client/Server Relationship

At a high level, an OPC server is comprised of several objects: the server, the group, and the item. The OPC server object maintains information about the server and serves as a container for OPC group objects. The OPC group object maintains information about itself and provides the mechanism for containing and logically organizing OPC items.

OLE for Process Control (OPC)

OPC groups provide a way for clients to organize data. For example, the group might represent items in a particular operator display or report. Data can be read and written. Exception-based connections can also be created between the client and the items in the group and can be enabled and disabled as needed. An OPC client can configure the rate at which an OPC server should provide data changes to the OPC client.

There are two types of groups, public and local (or 'private'). Public is for sharing across multiple clients, local is local to a client. Refer to the section on public groups for the intent, purpose, and functionality and for further details. There are also specific optional interfaces for the public groups. Within each group the client can define one or more OPC Items.

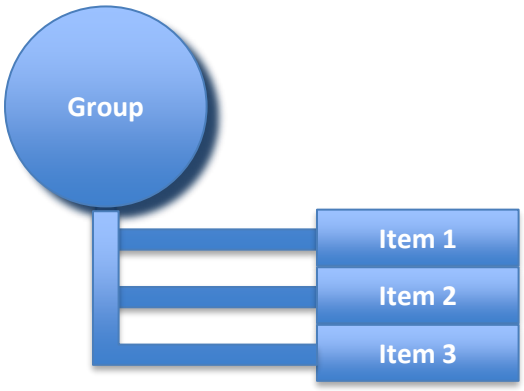


Figure #A.3 - Group/Item Relationship

The OPC items represent connections to data sources within the server. An OPC item, from the custom interface perspective, is not accessible as an object by an OPC client. Therefore, there is no external interface defined for an OPC item. All access to OPC items is via an OPC group object that contains the OPC item; in other words, where the OPC item is defined.

Associated with each item are a Value, Quality, and Time Stamp. The value is in the form of a VARIANT, and the Quality is similar to that specified by Fieldbus.

Note: Items are not the data sources. They are just connections to them. For example, the tags in a DCS system exist regardless of whether an OPC client is currently accessing them. The OPC item should be thought of as simply specifying the address of the data, not as the actual physical source of the data that the address references.

Where OPC Fits

Although OPC is primarily designed for accessing data from a networked server, OPC interfaces can be used in many places within an application. At the lowest level they can get raw data from the physical devices into a SCADA or DCS or from the SCADA or DCS system into the application. The architecture and design makes it possible to construct an OPC server that allows a client application to access data from many OPC servers provided by many different OPC vendors running on different nodes via a single object.

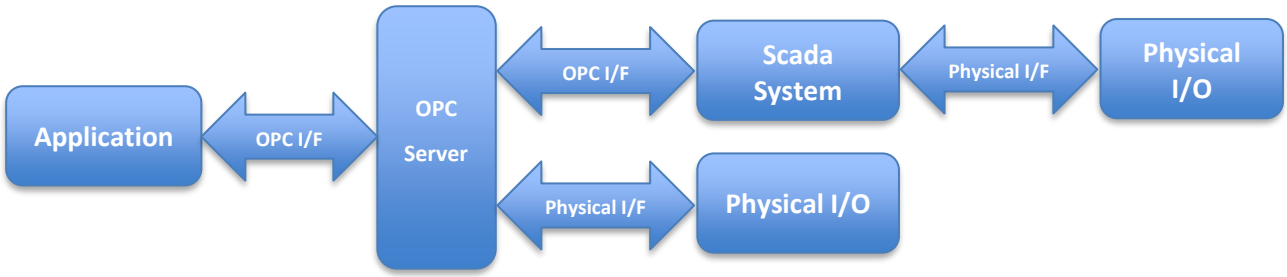


Figure #A.4

General OPC Architecture and Components

OPC is a specification for two sets of interfaces: the OPC Custom Interfaces and the OPC Automation interfaces. A revised automation interface will be provided with release 2.0 of the OPC specification. This is shown below.

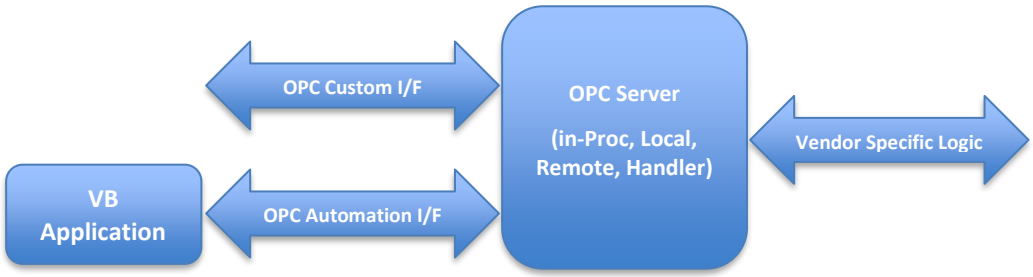


Figure #A.5

The OPC Specification specifies COM interfaces (what the interfaces are), not the implementation of those interfaces (that is, not the how of the implementation). It specifies the behavior that the interfaces are expected to provide to the client applications that use them. Included are descriptions of architectures and interfaces that seem most appropriate for those architectures. Like all COM implementations, the architecture of OPC is a client-server model where the OPC Server component provides an interface to the OPC objects and manages them. There are several unique considerations in implementing an OPC Server. The main issue is the frequency of data transfer over non-sharable communication paths to physical devices. Thus, we expect that the OPC Server will either be a local or remote EXE, which includes code that is responsible for efficient data collection from a physical device.

An OPC client application communicates to an OPC server through the specified OPC custom and automation interfaces. OPC servers must implement the custom interface, and optionally may implement the automation interface.

An inproc (OPC handler) may be used to marshal the interface and provide the additional item-level functionality of the OPC Automation Interface. Refer to the figure below:

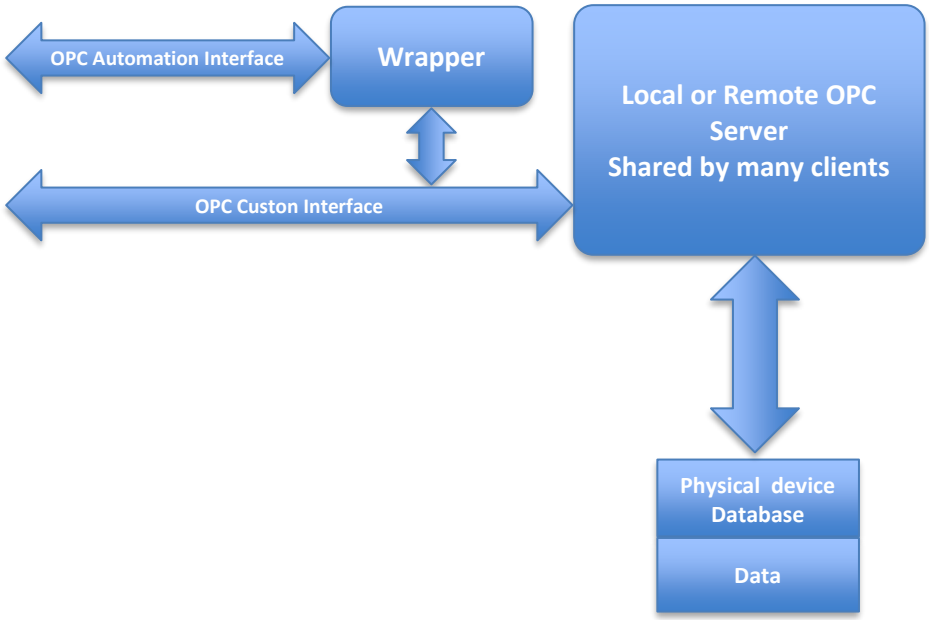


Figure #A.6 - Typical OPC Architecture

It is also expected that the server will consolidate and optimize data accesses requested by the various clients to promote efficient communications with the physical device. For inputs (Reads), data returned by the device is buffered for asynchronous distribution or **OLE for Process Control (OPC)** synchronous collection by various OPC clients. For outputs (writes), the OPC Server updates the physical device data on behalf of OPC Clients.

For more information please visit: http://www.opcfoundation.org/

APPENDIX B

ActiveX Technology

An ActiveX control is essentially a simple OLE object that supports an interface called IUnknown. It usually supports many more interfaces in order to offer functionality; but all additional interfaces can be viewed as optional and, as such, a container such as the ClearView Drawing Pad should not rely on any additional interfaces being supported. By not specifying additional interfaces that a control must support, a control can efficiently target a particular area of functionality without having to support particular interfaces to qualify as a control. As always with OLE, whether in a control or a container, it should never be assumed that an interface is available, and standard return-checking conventions should always be followed. It is important for a control or container to degrade gracefully and offer alternative functionality if a required interface is not available.

It is important for controls that require optional features, or features specific to a certain container, to be clearly packaged and marketed with those requirements. Similarly, containers that offer certain features or component categories must be marketed and packaged as offering those levels of support when hosting ActiveX Controls. It is recommended that controls target and test with as many containers as possible, and degrade gracefully to offer less or alternative functionality if interfaces or methods are not available. In a situation where a control cannot perform its designated job function without the support of a component category, that category should be entered as a requirement in the registry to prevent the control being inserted in an inappropriate container.

These guidelines define those interfaces and methods that a control can expect a container to support, although as always, a control should check the return values when using QueryInterface or other methods to obtain pointers to these interfaces. A container should not expect a control to support anything more than the IUnknown interface. These guidelines identify what interfaces a control can support and what the presence of a particular interface means.

Why Are ActiveX Controls Important?

ActiveX Controls have become the primary architecture for developing programmable software components for use in a variety of different containers (such as the ClearView Drawing Pad), ranging from software development tools to end-user productivity tools. For a control to operate well in a variety of containers, the control must be able to assume some minimum level of functionality that it can rely on in all containers.

By following these guidelines, control developers make their controls more reliable and interoperable, and, ultimately, better and more usable components for building component-based solutions.

For more information please visit: http://msdn.microsoft.com

APPENDIX C

OLE DB/ODBC

OLE DB Providers Overview

OLE DB architecture is built upon the precept of an application accessing diverse data stores through a small application built specifically for that purpose. The application that uses OLE DB functionality is called the consumer, while the one that accesses the data by exposing OLE DB interfaces is called the provider.

How does this relate to ClearView?

Providers fall into two categories: those providing services and those providing data. A service provider encapsulates a service by producing and consuming data through OLE DB interfaces. It does not own its own data and, in reality, serves a dual role of consumer and provider. A service provider may also be further defined as a service component, which must work in conjunction with other service providers or components. The Cursor Service for OLE DB is an example of this. A data provider owns its own data and exposes it in tabular form. It is not dependent on other providers—service or data—to provide data to the consumer. Providers can be simple or complex, depending on the needs of the consumer. They can return data intact or perform operations on the data. A set of mandatory interfaces is required to expose the minimal level of functionality supported by the data store. OLE DB providers are not required to support the complete OLE DB interface. Optional interfaces can be used for additional functionality.

Because providers are made up of COM components that contain a series of standard interfaces, consumers can access them using any programming language.

For more information please visit: http://msdn.microsoft.com/

ODBC Overview

The Microsoft® Open Database Connectivity (ODBC) interface is a C programming language interface that makes it possible for applications to access data from a variety of database management systems (DBMSs). The ODBC interface permits maximum interoperability—an application can access data in diverse DBMSs through a single interface. Furthermore, that application will be independent of any DBMS from which it accesses data. Users of the application can add software components called drivers, which interface between an application and a specific DBMS.

For more information please visit: http://msdn.microsoft.com/

APPENDIX D

Meeting 21 CFR Part 11 Requirements

ClearView-SCADA software offers components and tools to help the user in meeting the requirements of the 21 CFR part 11 regulations applicable to FDA validated processes. It will be assumed, in this specification, that these components and tools will only be used in "closed" systems. Digital signatures are used in "open" systems and so are not applicable in this specification.

Definition of a closed system: *A closed system is an environment in which access is controlled only by persons who are responsible for the content of electronic records of the system.*

Section	Requirement	ClearView
B/11.10	Controls for closed systems.	
	Persons who use closed systems to create, modify, maintain, or transmit electronic records shall employ procedures and controls designed to ensure the authenticity, integrity, and, when appropriate, the confidentiality of electronic records; and to ensure that the signer cannot readily repudiate the signed record as not genuine. Such procedures and controls shall include the following:	*The ClearView customer shall ensure electronic record authenticity, integrity and confidentiality.*
(a)	Validation of systems to ensure accuracy, reliability, consistent intended performance, and the ability to discern invalid or altered records.	The ClearView customer should reference any applicable validation protocols such as IQ or OQ.
(b)	The ability to generate accurate and complete copies of records in both human readable and electronic form suitable for inspection, review, and copying by the agency. Persons should contact the agency if there are any questions regarding the ability of the agency to perform such review and copying of the electronic records.	ClearView data can be exported in a human-and machine-readable format, as well as included in reports using Crystal Reports.
(c)	Protection of records to enable their accurate and ready retrieval throughout the records retention period.	The ClearView data is stored in a relational database and protected via system security. The customer shall establish policies to ensure that the records are retained for the appropriate duration of time.

Section	Requirement	ClearView
(d)	Limiting system access to authorized individuals.	ClearView user account information is stored in the ClearView system database. User passwords are encrypted. Ensuring the security and integrity of this database is the customer's responsibility.
(e)	Use of secure, computer-generated, time-stamped audit trails to independently record the date and time of operator entries and actions that create, modify, or delete electronic records. Record changes shall not obscure previously recorded information. Such audit trail documentation shall be retained for a period at least as long as that required for the subject electronic records and shall be available for agency review and copying.	ClearView logs all alarms and events in the Event Log. Additionally, tags can be configured to be audited, such that any user-initiated changes to the tag are recorded in the Event Log.
(f)	Use of operational system checks to enforce permitted sequencing of steps and events, as appropriate.	N/A
(g)	Use of authority checks to ensure that only authorized individuals can use the system, electronically sign a record, access the operation or computer system input or output device, alter a record, or perform the operation at hand.	ClearView uses a two-part User ID and password combination to authenticate a user. ClearView can be programmed to automatically log out a user after a period of inactivity.
B/11.10	Controls for closed systems.	
(h)	Use of device (e.g., terminal) checks to determine, as appropriate, the validity of the source of data input or operational instruction.	*The ClearView customer should reference any applicable validation protocols, e.g., the hardware IQ.*
(i)	Determination that persons who develop, maintain, or use electronic record/electronic signature systems has the education, training, and experience to perform their assigned tasks.	*The ClearView customer shall be responsible for training and training procedures.*
(j)	The establishment of, and adherence to, written policies that hold individuals accountable and responsible for actions initiated under their electronic signatures,	*The ClearView customer shall be responsible for developing policies and procedures to support the use of the*

Section	Requirement	ClearView
	in order to deter record and signature falsification.	*application in an FDA-regulated environment.*
(k)	Use of appropriate controls over systems documentation including:	
(1)	Adequate controls over the distribution of, access to, and use of documentation for system operation and maintenance.	The ClearView manuals are offered in a read-only format. Customers should ensure that operation and maintenance manuals are protected and maintained.
(2)	Revision and change control procedures to maintain an audit trail that documents time-sequenced development and modification of systems documentation.	The ClearView manuals are maintained under a revision control system.
B/11.30	Controls for open systems	
	Persons who use open systems to create, modify, maintain, or transmit electronic records shall employ procedures and controls designed to ensure the authenticity, integrity, and, as appropriate, the confidentiality of electronic records from the point of their creation to the point of their receipt. Such procedures and controls shall include those identified in § 11.10, as appropriate, and additional measures such as document encryption and use of appropriate digital signature standards to ensure, as necessary under the circumstances, record authenticity, integrity, and confidentiality.	*The ClearView customer shall be responsible for developing policies and procedures to support the use of the application in an FDA-regulated environment.*
B/11.50	Signature manifestations	
(a)	Signed electronic records shall contain information associated with the signing that clearly indicates all of the following:	
(1)	The printed name of the signer	Each ClearView Event Log entry includes the full printed name of the user.
(2)	The date and time when the signature was executed.	Each ClearView Event Log entry includes a time stamp of the event.

Section	Requirement	ClearView
(3)	The meaning (such as review, approval, responsibility, or authorship) associated with the signature.	Each ClearView Event Log entry includes a message with a description of the event.
(b)	The items identified in paragraphs (a)(1), (a)(2), and (a)(3) of this section shall be subject to the same controls as for electronic records and shall be included as part of any human readable form of the electronic record (such as electronic display or printout).	The ClearView Event Log can be exported in a human-and-machine-readable format, as well as included in reports using Crystal Reports.
B/11.70	Signature/record linking	
	Electronic signatures and hand-written signatures executed to electronic records shall be linked to their respective electronic records to ensure that the signatures cannot be excised, copied, or otherwise transferred to falsify an electronic record by ordinary means.	The ClearView Event Log is stored in a relational database with the printed user name as part of each entry. Database security to prevent unauthorized modification is the responsibility of the ClearView customer.
C/11.100	General requirements	
(a)	Each electronic signature shall be unique to one individual and shall not be reused by, or reassigned to, anyone else.	The ClearView User Manager prevents the creation of duplicate user IDs.
(b)	Before an organization establishes, assigns, certifies, or otherwise sanctions an individual's electronic signature, or any element of such electronic signature, the organization shall verify the identity of the individual.	The ClearView customer shall be responsible for verifying the identity of individuals using electronic signatures.
(c)	Persons using electronic signatures shall, prior to or at the time of such use, certify to the agency that the electronic signatures in their system, used on or after August 20, 1997, are intended to be the legally binding equivalent of traditional handwritten signatures.	*The ClearView customer shall be responsible for certifying the equivalence of electronic and traditional handwritten signatures.*
(1)	The certification shall be submitted in paper form and signed with a traditional handwritten signature, to the Office of	*The ClearView customer shall be responsible for certifying the equivalence of*

Section	Requirement	ClearView
	Regional Operations (HFC-100), 5600 Fishers Lane, Rockville, MD 20857.	*electronic and traditional handwritten signatures.*
(2)	Persons using electronic signatures shall, upon agency request, provide additional certification or testimony that a specific electronic signature is the legally binding equivalent of the signer's handwritten signature.	*The ClearView customer shall be responsible for certifying the equivalence of electronic and traditional handwritten signatures.*
C/11.200	Electronic signature components and controls	
(a)	Electronic signatures that are not based upon biometrics shall:	
(1)	Employ at least two distinct identification components such as an identification code and password.	ClearView uses a two-part User name and password combination to authenticate a user.
(i)	When an individual executes a series of signings during a single, continuous period of controlled system access, the first signing shall be executed using all electronic signature components; subsequent signings shall be executed using at least one electronic signature component that is only executable by, and designed to be used only by, the individual.	ClearView scripting has the ability to force user logout and login before a variable can be changed.
(ii)	When an individual executes one or more signings not performed during a single, continuous period of controlled system access, each signing shall be executed using all of the electronic signature components.	ClearView supports login time-out, after which the two-part user name and password must be re-entered to resume a session. The ClearView customer should also implement policies requiring a user to log out before leaving the workstation.
(2)	Be used only by their genuine owners	*The ClearView customer should implement policies to enforce non-sharing of user information.*
C/11.200	Electronic signature components and controls	

Section	Requirement	ClearView
(3)	Be administered and executed to ensure that attempted use of an individual's electronic signature by anyone other than its genuine owner requires collaboration of two or more individuals.	ClearView encrypts user passwords in the database to prevent easy discovery by the system administrator.
(b)	Electronic signatures based upon biometrics shall be designed to ensure that they cannot be used by anyone other than their genuine owners.	*The ClearView customer shall be responsible for the selection and usage of biometric devices.*
C/11.300	Controls for identification codes/pass-words	
	Persons who use electronic signatures based upon use of identification codes in combination with passwords shall employ controls to ensure their security and integrity. Such controls shall include:	
(a)	Maintaining the uniqueness of each combined identification code and password, such that no two individuals have the same combination of identification code and password.	The ClearView User Manager prevents the creation of duplicate user names.
(b)	Ensuring that identification code and password issuances are periodically checked, recalled, or revised (e.g., to cover such events as password aging).	ClearView enforces password aging.
(c)	Following loss management procedures to electronically de-authorize lost, stolen, missing, or otherwise potentially compromised tokens, cards, and other devices that bear or generate identification code or password information, and to issue temporary or permanent replacements using suitable, rigorous controls.	ClearView enforces account lockout.
(d)	Use of transaction safeguards to prevent unauthorized use of passwords and/or identification codes, and to detect and report in an immediate and urgent manner any attempts at their unauthorized use to the system security unit, and, as	ClearView audits login attempts to the Event Log. ClearView can generate an alarm for too many unauthorized attempts.

Section	Requirement	ClearView
	appropriate, to organizational management.	
(e)	Initial and periodic testing of devices, such as tokens or cards, that bear or generate identification code or password information to ensure that they function properly and have not been altered in an unauthorized manner.	*The ClearView customer is responsible for testing devices.*

APPENDIX E

ActiveX Grid Control - CVGRID

Alarm Time	ACK Time	Cleared Time	Alarm Name	Description	Severity	Group	User	Value
10/19/2017 9:32:11.85			ALARM_TEST	ALARM_TEST	WARNING	CVAlarm		11
			BUS5_TRIP	BUS5 TRIP	CRITICAL	BUS-5		
			Bus5_ZDZ_Alarm	Bus5 ZDZ Alarm	CRITICAL	BUS-5		
			Bus5_MTZ_Alarm	Bus5 MTZ Alarm	CRITICAL	BUS-5		
			Bus5_LDSH_Alarm	Bus5 LDSH Alarm	CRITICAL	BUS-5		
			BUS4_TRIP_G	BUS4 TRIP G	CRITICAL	BUS-4		
			Bus4_ZMH_Alarm	Bus4 ZMH Alarm	CRITICAL	BUS-4		
			Bus4_ZPN_Alarm	Bus4 ZPN Alarm	CRITICAL	BUS-4		
			Bus4_LDSH_Alarm	Bus4 LDSH Alarm	CRITICAL	BUS-4		
			Bus4_YPOB_Alarm	Bus4 YPOB Alarm	CRITICAL	BUS-4		

Figure 25.0

Properties

Property	Description	Example
AllowBigSelection	Returns/sets whether clicking on a column or row header should cause the entire column or row to be selected	Grid.AllowBigSelection = False/True
AllowUserResizing	Returns/sets whether the user should be allowed to resize rows and columns with the mouse	Grid.AllowUserResizing = False/True
Appearance	Returns/sets whether a control should be painted with 3-D effects	Grid.Appearance = False/True
BackColor	Returns/sets the background color of the CVGRID	Grid.BackColor = OLE Color
BackColorBkg	Returns/sets the background color of the CVGRID control	Grid.BackColorBkg = OLE Color
BackColorFixed	Returns/sets the background color of the fixed rows	Grid.BackColorFixed = OLE Color
BorderStyle	Returns/sets the border style for an object	Grid.BorderStyle = 0 (none) or 1 (border)
ColAlignment	Returns/sets the alignment of data in a column	Grid.ColAlignment(ColumnIndex) = 1 0 - The cell content is aligned left, top 1 - The cell content is aligned left, center (default) 2 - The cell content is aligned left, bottom 3 - The cell content is aligned center, top 4 - The cell content is aligned center, center 5 - The cell content is aligned center, bottom 6 - The cell content is aligned right, top 7 - The cell content is aligned right, center (default for numbers) 8 - The cell content is aligned right, bottom 9 - The cell content is of general alignment
Cols	Determines the total number of columns	Grid.Cols = 10
ColWidth	Determines the width of the specified column in Twips	Grid.ColWidth (2) = 2500
Edit	Enable Edit of the specific cell	Grid.Edit = False/True
Enabled	Returns/sets a value that determines whether an object can respond to user-generated events	Grid.Enabled = False/True

Property	Description	Example
FillStyle	Determines whether setting the Text property or one of the Cell formatting properties of a CVGRID applies the change to all selected cells	Grid.FillStyle = 0 0 - Single. Changing Text or any of the cell properties only affects the active cell. This is the default 1 - Repeat. Changing Text or any of the cell properties affects all selected cells
FixedCols	Returns/sets the total number of fixed (non-scrollable) columns	Grid.FixedCols = 1
FixedRows	Returns/sets the total number of fixed (non-scrollable) rows	Grid.FixedRows = 1
Font	Returns/sets the default font or the font for individual cells	Grid.Font = IFontDisp
ForeColor	Determines the color used to draw text on each part of the CVGRID	Grid.ForeColor = OLE Color
ForeColorFixed	Determines the color used to draw fixed row/column text of the CVGRID	Grid.ForeColorFixed = OLE Color
GridColor	Returns/sets the color used to draw the lines between FlexGrid cells	Grid.GridColor = OLE Color
GridColorFixed	Returns/sets the color used to draw the lines between fixed rows/columns of the CVGRID cells	Grid.GridColorFixed = OLE Color
GridLineWidth	Returns/sets the width in Pixels of the gridlines for the control	Grid.GridLineWidth = 1 (Integer)
MergeCells	Returns/sets whether cells with the same contents should be grouped in a single cell spanning multiple rows or columns	Grid.MergeCells = 1 (Integer) 0 - Never. The cells containing identical content are not grouped (default) 1 - Free. Cells with identical content always merge 2 - Restrict Rows. Only adjacent cells (to the left of the current cell) within the row containing identical content merge 3 - Restrict Columns. Only adjacent cells (to the top of the current cell) within the column containing identical content merge 4 - Restrict Both. Only adjacent cells within the row (to the left) or column (to the top) containing identical content merge
MergeCol	Returns/sets which columns should have their contents merged when the MergeCells property is set to a value other than 0 - Never	Grid.MergeCol (Index) = False/True
MergeRow	Returns/sets which rows should have their contents merged when the MergeCells property is set to a value other than 0 - Never	Grid.MergeRow (Index) = False/True
RowHeight	Returns/sets the height of the specified row in Twips	Grid.RowHeight (1) = 80
RowHeightMin	Returns/sets a minimum row height for the entire control, in Twips	Grid.RowHeightMin = 80
Rows	Determines the total number of columns or rows in a CVGRID	Grid.Rows = 100
ScrollBars	Returns/sets whether a FlexGrid has horizontal or vertical scroll bars	Grid. ScrollBars = 1 0 - The CVGRID has no scroll bars. 1 - The CVGRID has a horizontal scroll bar. 2 - The CVGRID has a vertical scroll bar. 3 - The CVGRID has horizontal and vertical scroll bars. (Default)

Property	Description	Example
TextMatrix	Returns/sets the text contents of an arbitrary cell	Grid.TextMatrix(Row, Col) = New_Text
WordWrap	Returns/sets whether text within a cell should be allowed to wrap	Grid.WordWrap = False/True
CellLeft	Returns the left position of the current cell, in twips	X = Grid.CellLeft
CellHeight	Returns the height of the current cell, in twips	X = Grid.CellHeight
CellWidth	Returns the width of the current cell, in twips	X = Grid.CellWidth
CellTop	Returns the top of the current cell, in twips	X = Grid. CellTop
FixedAlignment	Returns/sets the alignment of data in the fixed cells of a column	Grid.FixedAlignment (ColumnIndex) = 1 0 - The cell content is aligned left, top 1 - The cell content is aligned left, center (default) 2 - The cell content is aligned left, bottom 3 - The cell content is aligned center, top 4 - The cell content is aligned center, center 5 - The cell content is aligned center, bottom 6 - The cell content is aligned right, top 7 - The cell content is aligned right, center (default for numbers) 8 - The cell content is aligned right, bottom 9 - The cell content is of general alignment

Methods

Method	Description	Example
SortCol	Sorting by column	Grid.SortCol ColumnIndex, SortType 0 - None. No sorting is performed. 1 - Generic Ascending. An ascending sort, which estimates whether text is string or number, is performed 2 - Generic Descending. A descending sort, which estimates whether text is string or number, is performed 3 - Numeric Ascending. An ascending sort, which converts strings to numbers, is performed 4 - Numeric Descending. A descending sort, which converts strings to numbers, is performed 5 - String Ascending. An ascending sort using case-insensitive string comparison is performed 6 - String Descending. A descending sort using case-insensitive string comparison is performed 7 - String Ascending. An ascending sort using case-sensitive string comparison is performed 8 - String Descending. A descending sort using case-sensitive string comparison is performed 9 - Custom. This uses the Compare event to compare rows
Sort	Action-type property that sorts selected rows according to selected criteria	Grid.Sort SortType 0 - None. No sorting is performed. 1 - Generic Ascending. An ascending sort, which estimates whether text is string or number, is performed 2 - Generic Descending. A descending sort, which estimates whether text is string or number, is performed 3 - Numeric Ascending. An ascending sort, which converts strings to numbers, is performed 4 - Numeric Descending. A descending sort, which converts strings to numbers, is performed 5 - String Ascending. An ascending sort using case-insensitive string comparison is performed 6 - String Descending. A descending sort using case-insensitive string comparison is performed 7 - String Ascending. An ascending sort using case-sensitive string comparison is performed 8 - String Descending. A descending sort using case-sensitive string comparison is performed 9 - Custom. This uses the Compare event to compare rows
RemoveItem	Removes a row from a CVGRID control	Grid.RemoveItem RowIndex
SetRowBackColor	Sets the row background colors of individual cells	Grid.SetRowBackColor CInt(RowIndex), OLE_COLOR
SetColBackColor	Sets the column background colors of individual cells	Grid.SetColBackColor Cint(ColumnIndex), OLE_COLOR
SetCellBackColor	Sets the cell background colors of individual cells	Grid.SetCellBackColor Cint(RowIndex), Cint(ColumnIndex), OLE_COLOR
SetRowForeColor	Sets the row font colors of individual cells	Grid.SetRowForeColor CInt(RowIndex), OLE_COLOR
SetColForeColor	Sets the column font colors of individual cells	Grid.SetColForeColor Cint(ColumnIndex), OLE_COLOR
SetCellForeColor	Sets the cell font colors of individual cells	Grid.SetCellForeColor CInt(RowIndex), CInt(ColumnIndex), OLE_COLOR

Method	Description	Example
SetCellFont	Sets the cell font of individual cells	Grid. SetCellFont CInt(RowIndex), CInt(ColumnIndex), FontBold, FontItalic, FontName, CInt(FontSize), FontStrikeThrough, FontUnderline
Clear	Clears the contents of the CVGRID	Grid.Clear

Functions

Function	Description	Example
CellsBackColor	Returns/sets the background colors of individual cells or ranges of cells	X = Grid.CellsBackColor (OLE_COLOR)
CellsForeColor	Returns/sets the background colors of individual cells or ranges of cells	X = Grid.CellsForeColor (OLE_COLOR)
ColSels	Determines the starting or ending column for a range of cells (Read/Write)	X = Grid.ColSels (ColumnIndex)
RowSels	Determines the starting or ending row for a range of cells (Read/Write)	X = Grid.RowSels (RowIndex)
GetRowSels	Determines the starting or ending row for a range of cells (Read Only)	X = Grid.GetRowSels
SetGridCol	Sets the active cell in a CVGRID column (Read/Write)	X = Grid. SetGridCol (ColumnIndex)
SetGridRow	Sets the active cell in a CVGRID row (Read/Write)	X = Grid.SetGridRow (RowIndex)

Events

Event	Description	Example
Click	Fired when the user presses and releases the mouse button over the control	Private Sub Grid_Click ( ) 'Code here End Sub
DblClick	Fired when the user double-clicks the mouse over the control	Private Sub Grid_DblClick ( ) 'Code here End Sub
KeyDown	Fired when the user pushes a key	Private Sub Grid_KeyDown (KeyCode, Shift) 'Code here End Sub
KeyPress	Fired when the user presses a key	Private Sub Grid_KeyPress (KeyAscii) 'Code here End Sub
KeyUp	Fired when the user releases a key	Private Sub Grid_KeyUp (KeyCode, Shift) 'Code here End Sub
MouseDown	Fired when the user presses a mouse button over the control	Private Sub Grid_MouseDown (Button, Shift, x, y) 'Code here End Sub
MouseMove	Fired when the user moves the mouse over the control	Private Sub Grid_MouseMove (Button, Shift, x, y) 'Code here End Sub
MouseUp	Fired when the user releases a mouse button over the control	Private Sub Grid_ MouseUp (Button, Shift, x, y) 'Code here End Sub
SelChange	Fired when the selected range of cells changes	Private Sub Grid_SelChange ( ) 'Code here End Sub
RowColChange	Fired when the current cell changes	Private Sub Grid_ RowColChange ( ) 'Code here End Sub
EnterCell	Fired before the cursor enters a cell	Private Sub Grid_ EnterCell ( ) 'Code here End Sub
LeaveCell	Fired after the cursor leaves a cell	Private Sub Grid_ LeaveCell ( ) 'Code here End Sub

APPENDIX F

Comtrade Viewer ActiveX Control

The **Comtrade Viewer ActiveX Control** is high-resolution graphic interface for management, display and analysis of fault and periodic load Comtrade files (C37.111-1991/1999). The control supports the latest IEEE long file naming format (C37.232).

The advanced analysis interface offers a wide range of flexible analysis tools for navigating traces, peak hopping, amplitude and time scaling, superimposing, peak detecting, RMS and phase calculating, instantaneous values, primary and secondary values, sequence components, triggers, envelopes, merge files, append files and time synchronization of multiple records.

Computed Software Channels are also provided for advanced analysis. Software Channels can be used to compute +, - and 0 Sequence, Resistance, Envelopes, Missing Phases, Convert to Primary or Secondary Values, Current Imbalances, % of Rating...

Features:

- Phasors
- Harmonics
- Circular Charts
- Primary & Secondary Values
- Change Frequency
- Truncate Cycles
- Add Cycles
- User Defined Views
- Time Synchronization
- Frequency Calculator
- Complex Calculator
- Double Ended Fault Location
- Single Ended Fault Location
- Sequence Components Calculator
- Thiran Filter - Align Channel Data
- Super Impose Channels
- Software Analog Channels
- Convert to Comtrade
- IEEE Long Filenames
- SOE Tables on Multiple Records
- Fault Summaries
- File Sort & Queries
- Specialized Copy/Move
- ASCII & Binary Editors
- Save Sample Values
- Disturbance Reports
- Append Multiple Files
- Merge Multiple Files
- Adjust Files Time
- Play Channels Audio

Activate Comtrade Viewer ActiveX Control

- Open ClearView-SCADA Client
- From Menu Select Help | Activate COMTRADE Viewer

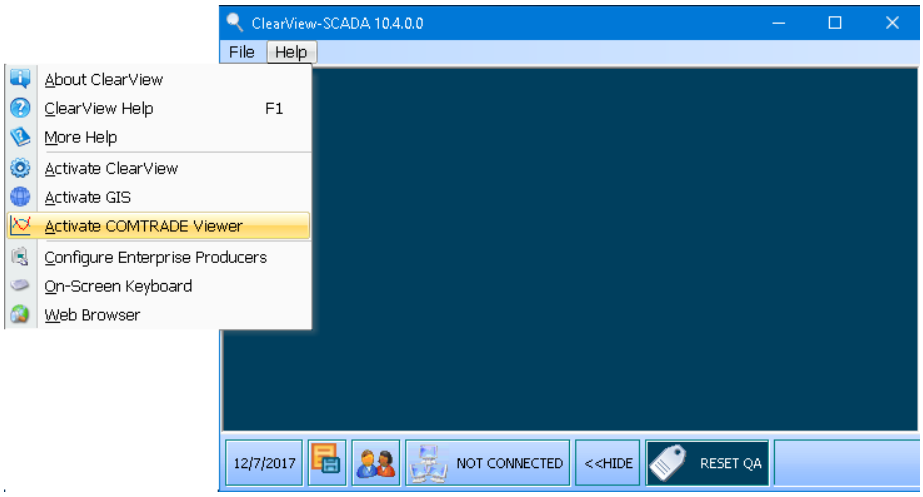


Fig A

- Copy **Customer ID** and provide this code to software provider

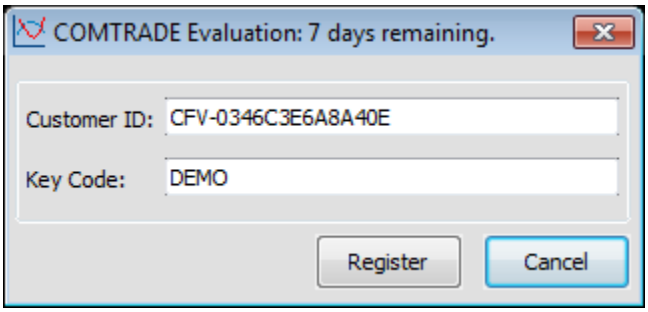


Fig B

- Enter key code (license) into **Key Code** entry box and click register button

Note: 7 days evaluation period is provided

Adding Comtrade Viewer ActiveX Control to ClearView-SCADA Screen

- In Design view select **Comtrade Viewer** object from **Data Controls** in **Object Library** and place the object on screen

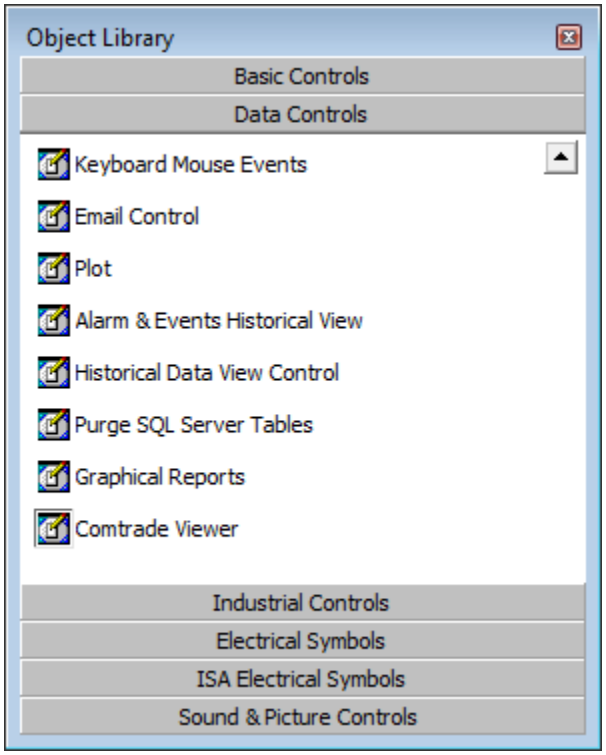


Fig C

- The following **Comtrade Viewer Control** properties are available
  - Font
  - MyProp – path to Comtrade file

C H A P T E R 1

Analysis Features

The Analysis Display offers a high-resolution graphical interface for displaying, analyzing, and manipulating analog and digital channels of an oscillography record or a periodic load file. Displayed channels can be marked, moved, zoomed, removed, restored, superimposed, scaled, numerically processed, exported and summarized.

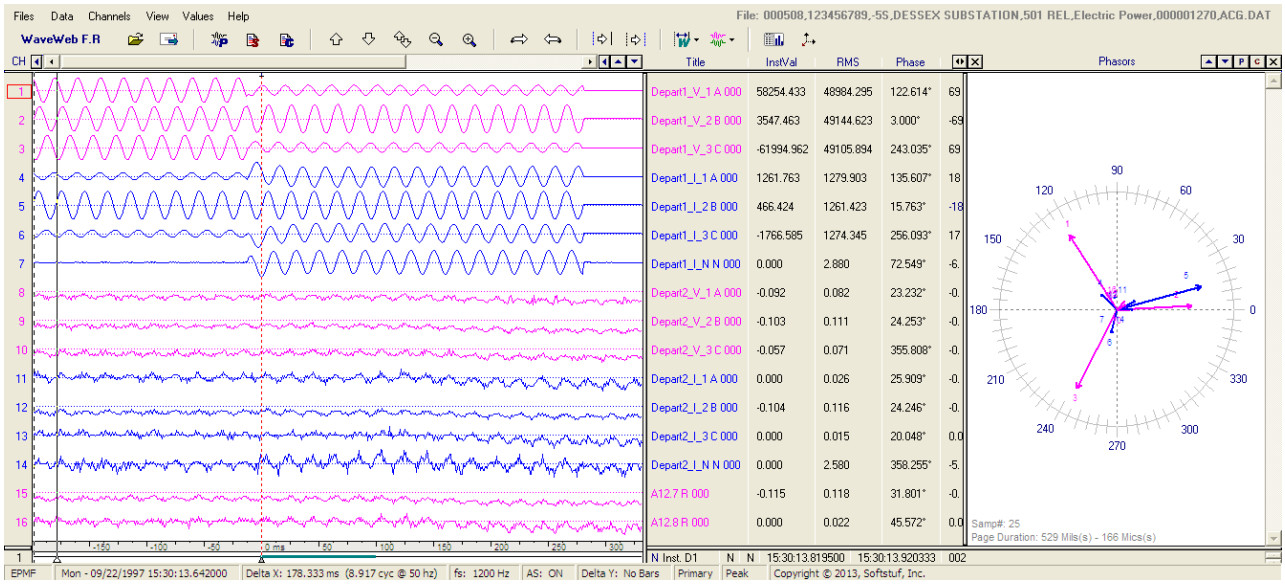


Figure 1.1 Analysis Display

The Analysis Display contains two sections: the analog view and the digital view. The analog view plots the oscillography or load data. The analog table displays values such as the channel's highest peak, RMS, phase, reference, instantaneous, maximum, and minimum values. The cursor bars are used to view the data values. The digital view plots the events and sensors and displays the channel's original state, the channel's final state, time of the first change, time of the last change, and the number of times the channel changed state.

Up to 256 analogs and 1024 digital channels can be displayed. The main features are described below.

Phasors

The phasor diagram shows a vector for each visible analog channel. The diagram is displayed to the right of the analog information window. To increase or decrease the size of the phasor window place the cursor over the vertical separator between the analog information window and the phasor window and drag the mouse to the left to increase or to the right to decrease. To close the phasor window, click the close button located in the header. To navigate the phase angles use the left arrow, right arrow, home, end, page up and page down keys or the data scroll bar. To increase/decrease the length of a channel's vector, mark the channel and use the increase/decrease amplitude menu buttons or the Ctrl-Up and Ctrl-Down keys. To increase/decrease only the length of the vectors, use the up and down phasor buttons. 

To toggle between the phasor display and the circular chart display click the "P" button above the phasor display for phasors or the "C" button for a circular chart. 

There are two types of phasor displays: non-referenced and referenced. The non-referenced display shows the phase angle for each sample in the display. The reference display shows the phase angle for each sample with respect to the

reference channel. The reference channel is the first marked channel in the window. All angles at a sample are subtracted from the reference angle. If there are no marked channels the non-referenced display is shown.

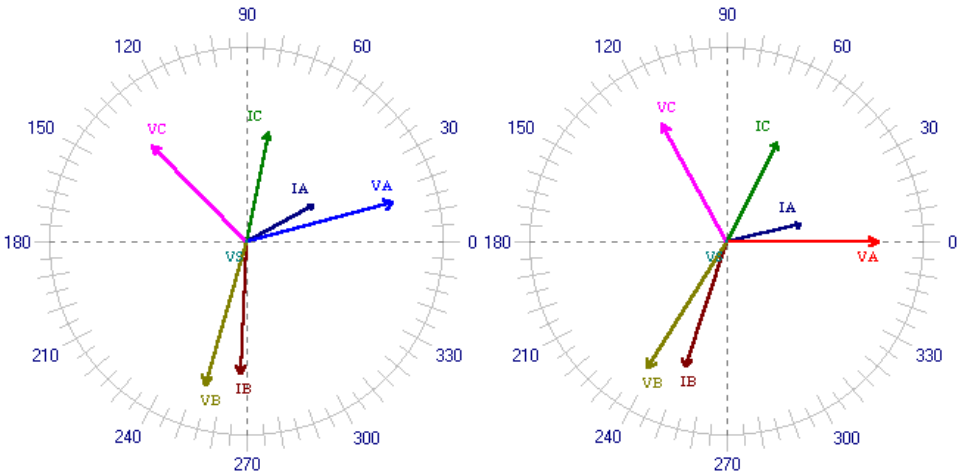


Figure 1.2 (a) Non-Referenced Phasors (b) Referenced Phasors

Harmonics

The harmonics window displays as many harmonics as possible based on the file's sampling frequency. A maximum of 200 harmonics can be displayed in the table. To display the harmonics window, click on the  menu button or select the Harmonics table menu option under the "View" menu. The harmonics window displays the first marked analog channel or if no channels are marked, the first visible channel. Changing the marked channel in the data plotting window will update the harmonics window with the appropriate channel.

The harmonic calculation is performed on a one cycle window, starting at the RMS bar and going forward to the data bar. There are three fields displayed at the bottom of the harmonics table and histogram; TrueRMS, CalculatedRMS and Total Harmonic Distortion (THD). The TrueRMS field displays the RMS value calculated by using the samples in the active cycle displayed in the waveform trace window. The CalculatedRMS field displays the square root of the summation of the squares of the DFT Magnitudes from harmonics 2 to the maximum harmonic divided by square root of 2. The THD field displays the square root of the summation of the squares of the DFT Magnitudes from harmonics 2 to the maximum harmonic and that quantity divided by the DFT Magnitude of the Fundamental.

The harmonics can be viewed in a table format or in a histogram. Click on the harmonics toggle button  to change the view. The histogram can show only one column from the table. To change the data displayed click the histogram drop down menu and select the column. The default view is the % of Fundamental.

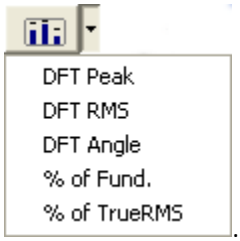


Figure 1.3 Histogram Drop Down Menu

The harmonic histogram bars can be resized using the resize up and down arrows 30 to display more or less harmonics in the window. The text displayed above the histogram bars can be shown or hidden be clicking on the Show/Hide text bar button . The harmonics window can be resized by dragging the edge of the window to show more or less harmonics per window.

Also, a vector for each harmonic is displayed in the phasor diagram. To hide/show the harmonic vectors toggle the "Vector Harmonics" menu option under the "View" menu from checked=on to unchecked=off.

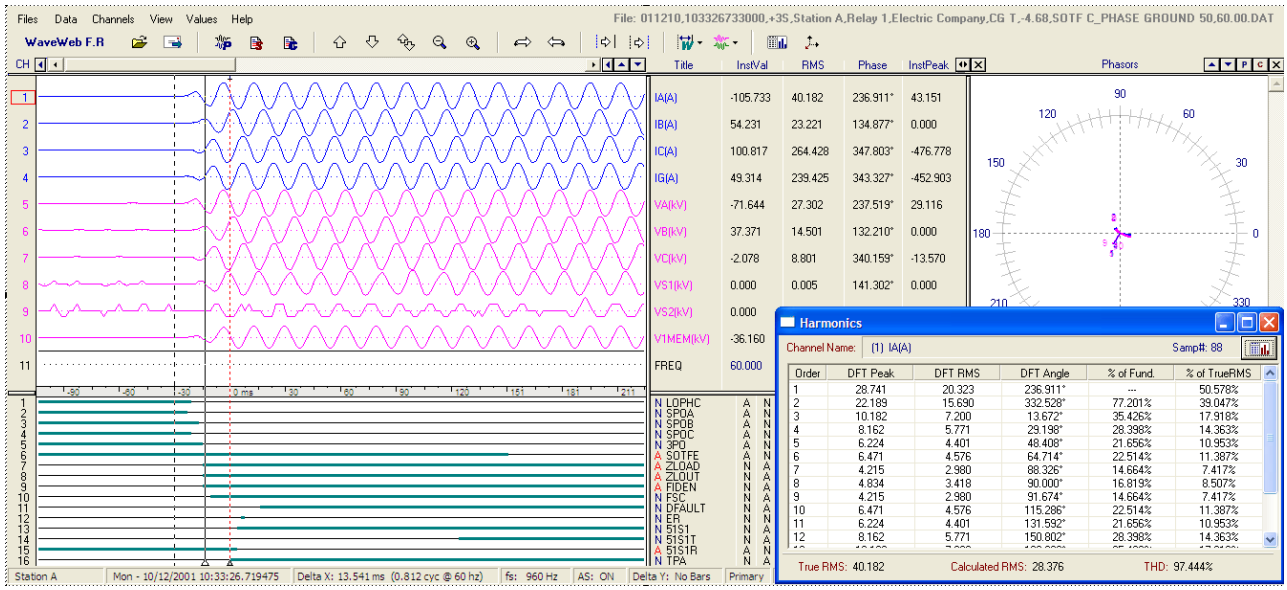


Figure 1.4 Harmonics Table View

Circular Chart

The Circular Chart diagram shows a circular display for each visible channel. The diagram is displayed to the right of the analog information window. The amount of data displayed in the circular chart is equal to the amount of data displayed in the waveform trace window. The duration of the data displayed is shown at the bottom of the circular chart. To increase or decrease the size of the circular chart window place the cursor over the vertical separator between the analog information window and the circular chart window and drag the mouse to the left to increase or to the right to decrease. To close the circular chart window, click the close button ☒ located in the header.

To navigate the circular chart, use the left arrow, right arrow, home, end, page up and page down keys or the data scroll bar. The cursor bar on the circular chart displays where the data bar is in the chart. To increase/decrease the display area of a channel on the circular chart, mark the channel and use the increase/decrease amplitude menu buttons or the Ctrl-Up and Ctrl-Down keys or use the up and down arrow button located in the circular chart header.

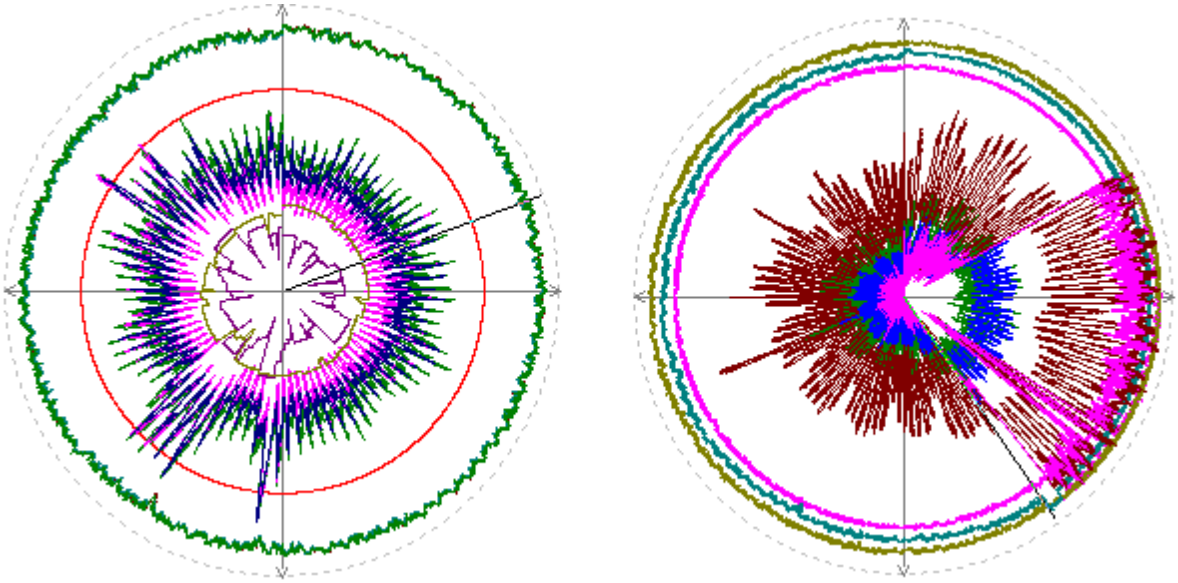


Figure 1.7 Circular Charts

Display Driver's Data Type

The data stored in the displayed file can be instantaneous values or RMS values. The default setting for all drivers is instantaneous values. If the Comtrade file saves the sample values as RMS calibrated, set the display driver's data type to RMS Type (Root 2 Multiplier). If the display driver is not set properly the analog column data will be displayed incorrectly. To set the driver's data type click the "Window Properties" menu button from the speed bar or select the "Window Properties" option under the "File" menu. Click the "Driver Data Type" tab and set the "Display Device's Data Type" field to "RMS Type" for RMS calibrated values and "Peak Type" for instantaneous values.

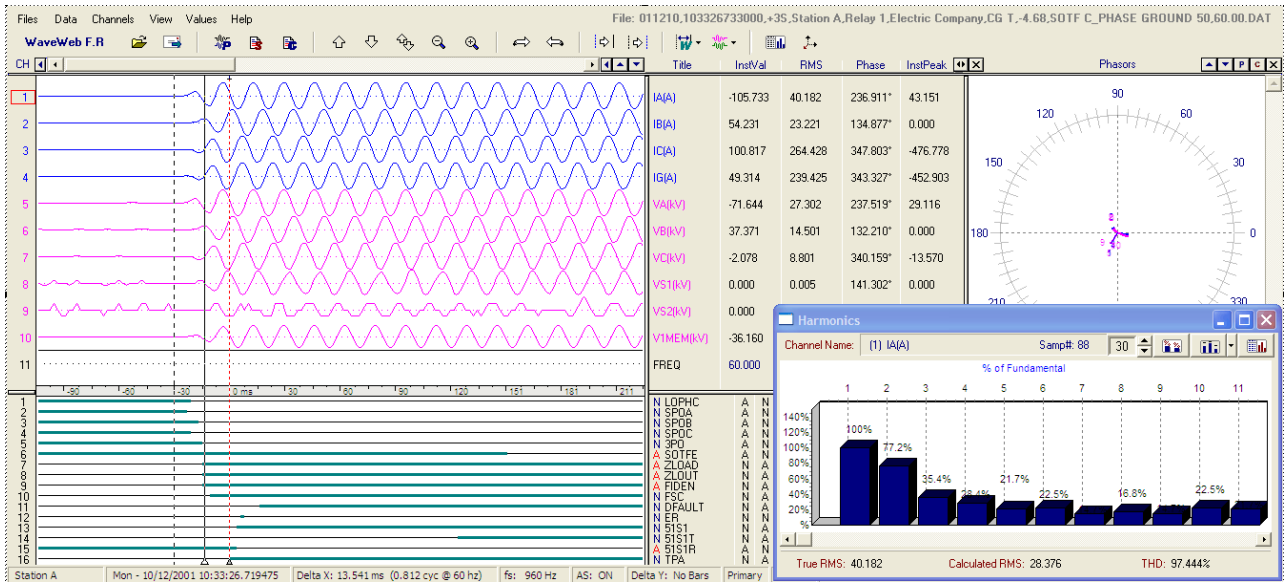


Figure 1.5 Harmonics Histogram View

Periodic Log Files

The periodic log viewer allows for viewing and analyzing large amounts of event data in a single display. The data is displayed in envelope form and may contain one day, one week, one month or one year of event data. This feature is useful for load flow analysis.

A circular chart of the data displayed in the trace window is plotted to the right of the channel information window. The circular chart cursor is positioned on the sample at the waveform data bar. The duration of the data displayed also is shown below the circular chart along with the sample number at the cursor bar.

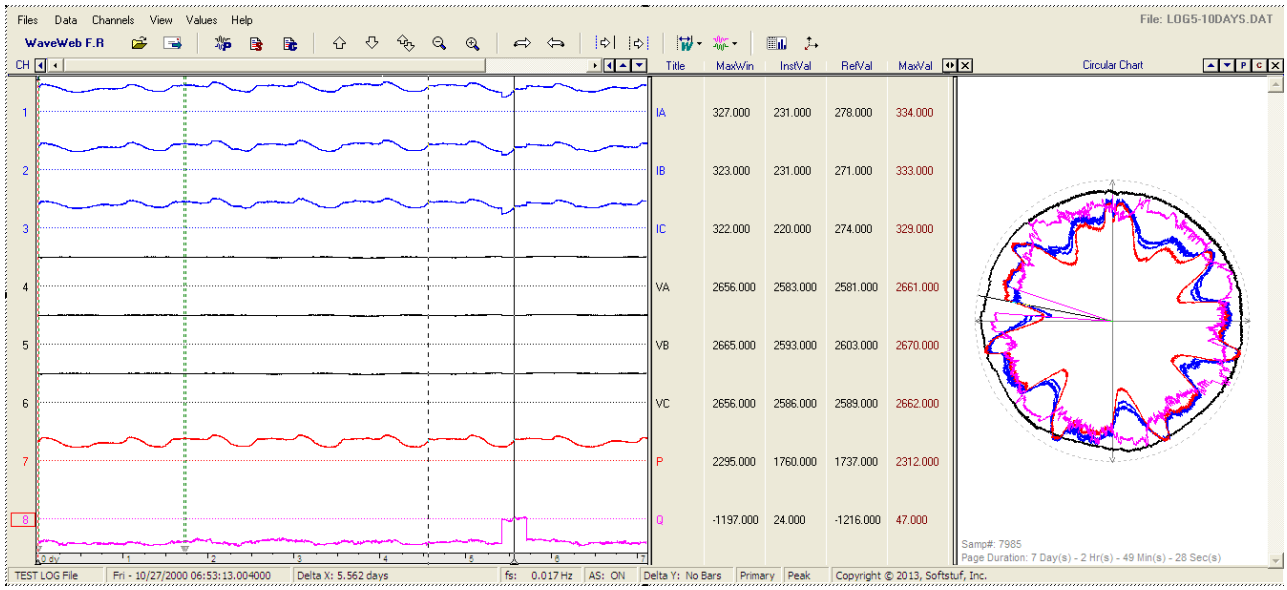


Figure 1.6 Periodic Log File

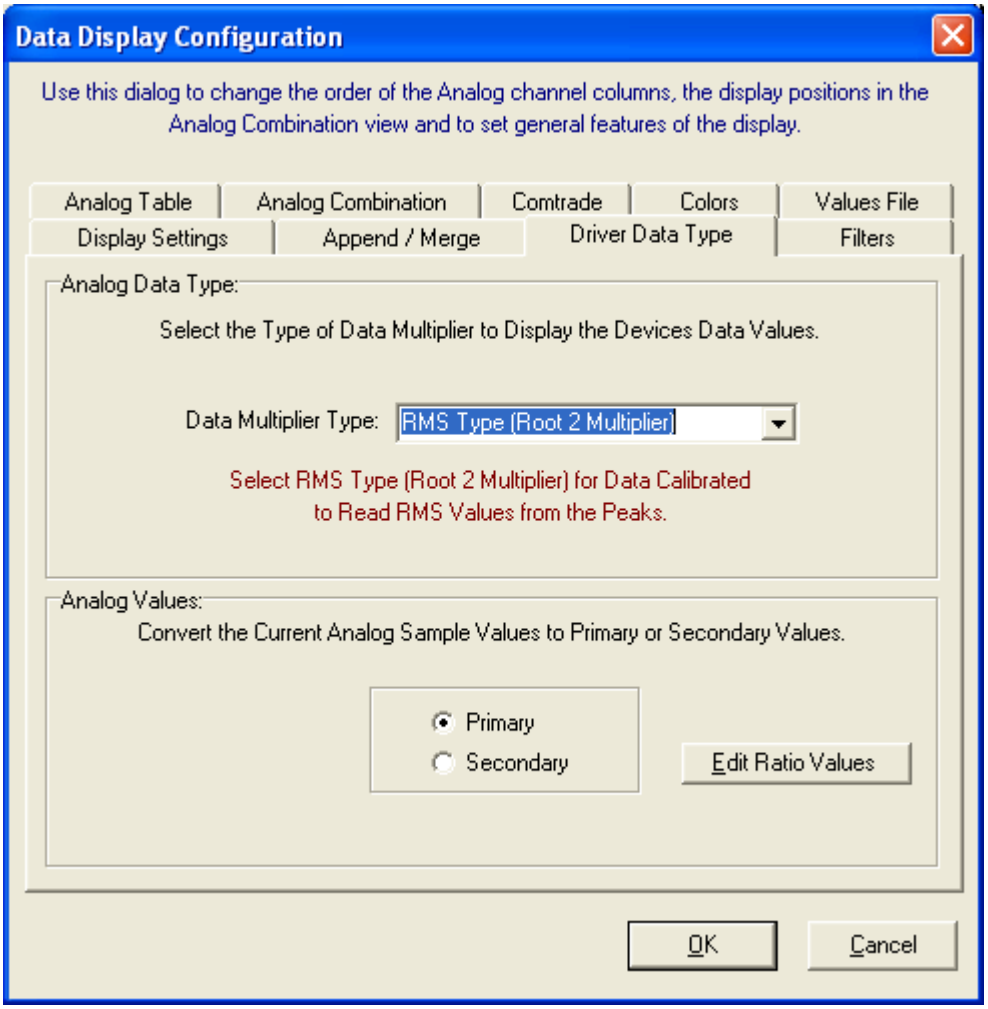


Figure 1.8 Analog Data Type Setting

Open Waveform File

To open a waveform file, click the open menu button  and select some files from the dialog.

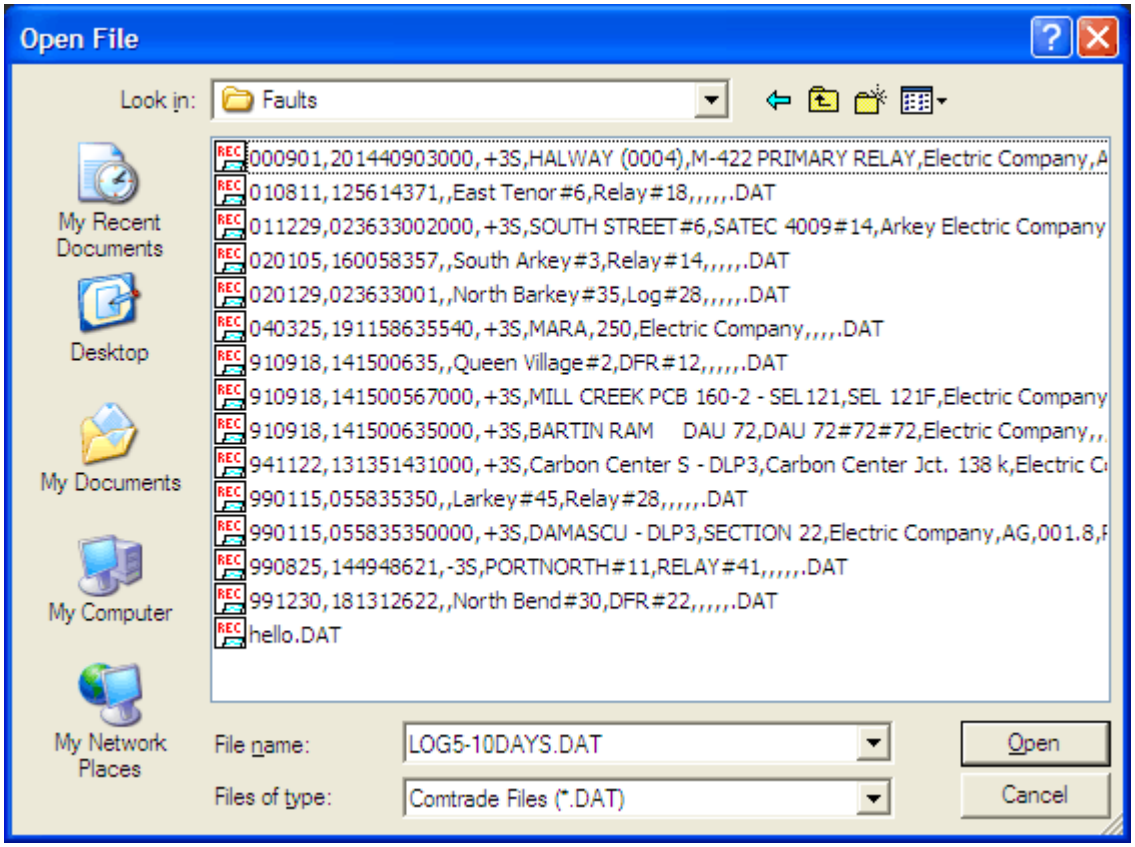


Figure 1.9 Open Data File

Email Active File

The active data file can be emailed by clicking on the email menu button or by selecting “Email Active File” from the “File” menu. The data file displayed and all the support files associated with the file are included as attachments. The users default email program will be displayed to email the file(s).

Navigating

Use the up and down arrow keys or the vertical scroll bar to browse the analog channels. The tab key toggles between the analog and digital views.

The analog and digital values are displayed in a table to the right of the channel traces. Hold the mouse button down and drag the table separator bars to resize the viewing area. To view the analog sample values, use the following navigation tools:

Left and right arrow keys to navigate sample by sample

Ctrl+left or ctrl+right keys to peak navigate

Shift+ctrl+left or shift+ctrl+right keys to cycle hop

Home and end keys to display the channel’s first and last samples

Triangle ⚠ at the bottom of the data bar to drag the data bar through the samples

Page up and page down keys to page through the samples

Left button displayed to the left of the data scroll bar to move the sample at the data bar to the position of the first sample displayed

Left button displayed to the right of the data scroll bar to move the trace and table separator to the position of the data bar

Click the left/right arrow button (located to the right of the analog table headers) or use the shift-right/left arrows to scroll through the analog table columns. Refer to the "Viewing Analog Data" section for field descriptions.

NOTE: If no channels are marked then the peak navigate and cycle hop features navigate through the first channel's data.

Setting the Cursor Bars

Four vertical cursor bars are displayed in the analog view. The blue dotted line represents the reference bar, the black solid line represents the data bar, the black dotted line represents the RMS bar and the red dotted line represents the fault position defined in the file's configuration information. There are also two horizontal bars displayed when the "Horizontal Bars" menu option under the "View" menu is checked.

Data Bar

The data bar is displayed as a black solid line with a white triangle below the line. The data bar is automatically displayed at the end of the first cycle in the data window when it is first displayed. To move the data bar, use the left and right arrow keys to move one sample, use the Ctrl-left and Ctrl-right keys to peak hop, use the Shift-Ctrl-left and Shift-Ctrl-right keys to cycle hop, use the page up and page down keys to move one page up or down or left click the mouse to move to any position in the data or drag the triangle to scroll through the data. When the mouse is held over the triangle a hint message displays the sample number at the data bar and the delta time from the first sample. The time of the sample at the data bar is displayed in the second status bar field. The channel values at the data bar are displayed to the right of the traces in the analog channel information table.

Reference Bar

The reference bar is displayed as a blue dotted line. The reference bar is defaulted to the fault time specified in the file. To move the reference bar to the position of the data bar use the "Move Reference Bar to Data Bar" option inside the "View" menu or press Ctrl-A or click the **SetRef** menu button . Click the opposite mouse button to move the reference bar to any position in the data area. The status field Delta X in the status bar at the bottom of the screen shows the time difference (in milliseconds or seconds) between the reference bar and the data bar. It also shows how many cycles are between the two bars.

RMS Bar

The RMS bar is displayed as a black dotted line. The RMS bar is defaulted to one cycle away from the data bar, except when the data bar is positioned at the beginning of the data. This bar is used for calculating the RMS value displayed in the analog information table. The RMS value in the analog table is calculated using all of the sample values displayed between the data bar and the RMS bar. To move the RMS bar to the position of the reference bar (blue dotted line) use the "Move RMS Bar to Reference Bar" option inside the "View" menu or press Ctrl-Z or click the **SetRMS** menu button.

Fault Bar

The fault bar is displayed as a red dotted line. The fault bar is fixed and positioned at the fault time defined in the file's configuration information. The fault bar can be shown or hidden by selecting the "Yes" or "No" options for the "Show Vertical Fault Bar" field in the properties dialog under the "Display Settings" tab.

Horizontal Bars

When the “Horizontal Bars” menu option under the “View” menu is checked two horizontal bars will be displayed. The solid black line follows the data bar and the dotted blue line follows the reference bar. The bars will be positioned at the first marked analog channel (displayed in red), if no channels are marked, then they are positioned at the first displayed channel. The Delta Y field in the status bar shows the difference between the two bars.

To automatically resize the RMS sliding window click on the **Resize Sliding Window** menu button . To manually resize the RMS sliding window click the opposite mouse button to set the reference position and the mouse button to set the data bar position then click the **SetRMS** menu button. The RMS bar is moved to the reference position. The Delta X field displayed in the status bar at the bottom of the screen shows the time difference (in milliseconds or seconds) between the date bar and reference bar and the number of cycles between the two bars. Use the left, right, ctrl+left, and ctrl+right, shift+ctrl+left, and shift+ctrl+right keys or the horizontal scroll bar to move the sliding window.

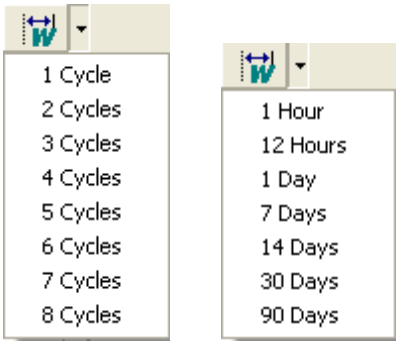


Figure 1.10 Resize Sliding Windows Drop Down Menus

Marking, Deleting, and Restoring Channels

To mark or unmark a channel, mouse click on the channel ID or channel title, or use the space bar. To mark a group of channels click on the first channel then shift click on the last channel. Marked channels are displayed in red.

To mark/unmark all analog and digital channels select the “Mark/UnMark All” menu option under the “Channels” menu option. If no channels are marked, all of the analog and digital channels will be marked. If any channels are marked, all of the channels will be unmarked. To mark/unmark all the analog channels select the “Analog Mark/Unmark All” menu option under the “Channels” menu. To mark/unmark all of the digital channels select the “Digital Mark/Unmark All” menu option under the “Channels” menu.

Channels must be marked to delete them from the display window. The Delete key removes the marked channels and the Insert key restores all the deleted channels.

Scaling Analog Channels

When the analysis display is initially opened, all the analog channels are scaled to one value. To scale the channels according to the maximum space allocated for display, click the **Auto Scale** menu button. This option toggles among the three views: On, Off and ++. The active auto scale state is displayed in the “AS” status field. Each view is defined below:

- **ON** – The On view plots the channel data scaled to the maximum value allocated along the zero-reference line.
- **OFF** – The Off view plots all of the channels that are scaled to the maximum value in the file.

- **++** - The ++ view plots the signal using the number of maximum pixels allocated for the channel. The highest value is plotted at the maximum position and the smallest value is plotted at the lowest position. This feature was added to clearly show the profile of a frequency channel, a Vdc channel and load data channels.

To increase or decrease a channel's amplitude, along with the phasors and circular chart first mark the channels then click the **AmpUp** or **AmpDn** menus buttons or use the ctrl+up/down arrow keys. The auto scale multiplier (ASM) is used to amplify or attenuate the channel's data values. For example, when the amplitude increases the ASM value is multiplied by the channel's current "Pixsdisp" and when the amplitude decreases the ASM value is divided by the channel's current "Pixsdisp". To change the ASM value, select the "Properties" menu option under the "File" menu then click the "Display Settings" tab, enter a number and click **OK**. This value is initially defaulted to 2.00. To increase/decrease only the analog channels amplitude, click the up and down arrow buttons located to the right of the data scroll bar. To increase/decrease only the phasor magnitude or circular chart click the up and down arrow buttons located in the phasor/circular chart header.

To increase or decrease the channel's time scale, click the **Condense** or **Expand** menu buttons or press the ctrl+page up and ctrl+page down keys.

NOTE: If no channels are marked all the visible channels are scaled accordingly.

Zooming Channels

To zoom in on specific analog or digital channels, first mark the channels then press <enter> or click the **ViewMrks** menu button. The unmarked channels are removed from the display window and the marked channels are rescaled to fit in the window. To restore the hidden channels press the <esc> key, the <backspace> key, or click the **ViewAll** menu button.

When returning to the original view all channels in the previous view remain marked for quick selection of additional channels for a new view.

Repositioning Channels

Analog channels can be repositioned in the display window. To move an analog channel up one position mark the channel and press the "+" key or select "Shift Marks Up" from the "Channel" menu. To move a channel down one position mark the channel and press the "-" key or select "Shift Marks Down" from the "Channel" menu. To move a group of analog channels, mark all the desired channels then press the "+" or "- "keys.

Saving as

The visible analog and digital channels can be saved in the COMTRADE ASCII or Binary format or in a CSV format. The Comtrade versions supported are: the 1991 and 1999 format. The CSV formats supported are: RMS Values, Instantaneous Values, Vector Values (Mag &Ang or RMS & Ang).

Mark the analog and digital channels to save and press <enter> or click the **ViewMrks** menu button . To save the channel values file select the "Save as menu option from the "File" menu. Enter the destination path and filename (do not define the filename extension) and select the Save Type then click **OK**.

To automatically name the file using the IEEE long file naming format check the "Use the ComNames Naming Convention to Name the Comtrade File(s)" field in the "Save As" dialog and leave the "File Name" field empty. The selected channels are converted to the specified format and are named using the IEEE long file naming convention.

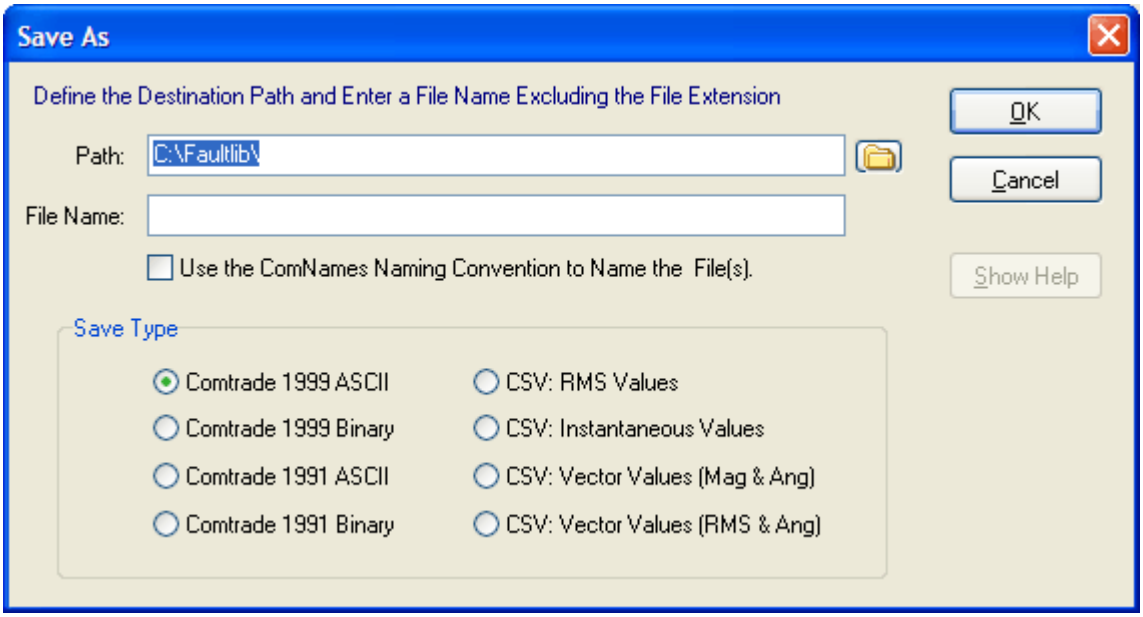


Figure 1.11 Save As

Viewing Analog Data

The values displayed in the analog view can be presented in tabular form (analog table) or in a concentrated form (combination view). Press F4 to toggle between the two views. The concentrated view can only be displayed if there is enough room to display 2 lines of data values between each channel. To navigate through the analog table columns use the View button (located to the right of the analog table headers) or the shift-right/left arrow keys. To close the analog table, click the Close button located in the header. Valid analog channels are displayed in the left portion of the window and the analog information in the center table. An analog channel is marked as invalid if the title is empty, or it contains any of the following strings in the beginning of the title.

- UNUSE
- UNDEF
- NOT D
- NOT U
- NOT I
- NAT A
- UNDEF
- {
- N/A
- ANALOG INPUT
- ANALOG CHANNEL
- EXTERNAL INPUT
- EVENT CHANNEL
- CHANNEL
- DIGITAL TRACE #
- SPARE

A maximum of 256 analog channels can be displayed in one window. The values displayed in the analog table and combination view are described below.

Analog Table View:

The analog table view is the default view. Use the view button or the shift-right/left arrow keys to navigate through the columns of the table. The original sample values are plotted according to one of the following data types:

- Peak to Peak data
- RMS Calibrated data
- Log files.

All of the display drivers in the system are defaulted to peak to peak except the predefined log drivers. To change the settings for a driver select the "Window Properties" option under the "File" menu. Click on the "Driver Data Type" tab and select the type from the "Data Multiplier Type" drop down list. Periodic Log File's data type cannot be changed.

The following tables describe the analog data for the sinusoidal peak-to-peak, non-sinusoidal, and sinusoidal RMS data types:

Peak to Peak

Field	Description
Title	The analog channel titles.
RMS	The TrueRMS value is calculated by taking the summation of the square of all the sample values that are between the RMS bar (black dotted line) and the data bar. The result is divided by the total number of samples between the two bars and takes the square root of that result.
InstPeak	The highest absolute value of all of the samples between the two zero reference crossings surrounding the data bar (black solid line).
Phase	The phase angle of each channel.
InstVal	The sample value at the data bar (black solid line).
RefVal	The sample value at the reference bar (blue dotted line).
MaxPeak	The maximum peak value of the channel.
MinPeak	The minimum peak value of the channel.
Units	The analog channels prefix and units.
PixsDisp	The number of pixels allocated for displaying the trace.
DFT Peak	The DFT Magnitude calculated between the RMS bar (black dotted line) and the data bar (solid data bar).
Crest	The DFTMag column divided by the RMS column.

Sinusoidal RMS Calibrated

Field	Description
Title	The analog channel titles.
RMS	The RMSVal column calculates an RMS value for all of the samples between the RMS bar (black dotted line) and the data bar (black solid line). Since the data is RMS calibrated each sample value is multiplied by the square root of 2 before it is squared.
InstPeak	The square root of 2 multiplied by the peak value measured between the two reference crossings surrounding the data bar (black solid line).
Phase	The phase angle of each channel.
InstVal	The RMS sample value at the data bar (black solid line) multiplied by Root 2.

Field	Description
RefVal	The RMS sample value at the reference bar (blue dotted line) multiplied by Root 2.
MaxPeak	The RMS maximum peak value of the channel multiplied by Root 2.
MinPeak	The RMS minimum peak value of the channel multiplied by Root 2.
Units	The analog channels prefix and units.
PixsDisp	The number of pixels allocated for displaying each trace.
DFT Peak	The DFT Magnitude calculated between the RMS bar (black dotted line) and the data bar (solid data bar).
Crest	The DFTMag column divided by the RMS column.

Non-Sinusoidal (Load Files)

Field	Description
Title	The analog channel titles.
MaxWin	The absolute maximum value between the sliding window bar (black dotted line) and the data bar (black solid line).
InstVal	The sample value at the data bar (black solid line).
RefVal	The sample value at the reference bar (blue dotted line).
MaxVal	The maximum value of the channel.
MinVal	The minimum value of the channel.
Units	The analog channels prefix and units.
PixsDisp	The number of pixels allocated for displaying the trace.
AvgWin	The average value of all of the samples between the sliding window bar (black dotted line) and the data bar (black solid line)

Combination View:

The combination view shows all of the channel information in a signal view. This view is only available if there is sufficient room between analog channels to display two or more lines of text.

Default Display format:

Peak to Peak:

Channel Title			
RMS	MaxPeak	RefVal	
InstVal	MinPeak	Units	ASV

RMS Calibrated:

Channel Title			
RMS	MaxPeak	RefVal	
InstVal	MinPeak	Units	ASV

Load Files:

Channel Title			
MaxWin	MaxVal	RefVal	
InstVal	MinVal	Units	ASV

The peak sample values are displayed in red when the data bar is on the channel's maximum value and displayed in blue when the data bar is on the channel's minimum value. Use the Tab key to toggle between the analog and digital channels. To hide the channel information, select the "Channel Information" menu option from the "View" menu.

The analog table and combination views can be resized by selecting the vertical separator bar and dragging it to the right or left. The mouse icon changes to the vertical resize cursor when the mouse is positioned over the separator bar.

To change how the analog data is displayed in the analog table and combination view select the "Properties" menu option from the "File" menu or click on the "Properties"  menu button from the speed bar. The "Analog Table" tab and the "Analog Combination" tab allows for changing the appearance of the analog information window.

Some of the functions of the "Properties" dialog is reordering, hiding, and showing the analog table columns; changing the data positions in the combination view; changing the background colors and trace colors; and for changing the driver's data type and trace/phasor scale multipliers.

Viewing Digital Data

The default digital view consists of only the triggered digital channels, which are displayed at the bottom of the screen. To view all of the digital channels including the unused channels select "All Digital Channels" from the "View" menu.

The digital trace is displayed as a thin black line when the sample value equals the original state defined in the displayed format and is displayed as a thick green line when the sample value differs from the original state. The Cursor State column in the digital table displays an "A" for Alarm and "N" for Normal. These values are set by comparing the sample value at the data bar with original state, "A" = different than original state, "N" = same as original state.

The digital information is displayed in tabular form. The data columns are described below:

Column Number	Description
1 – Cursor State	The digital state of the sample at the cursor position (A=Alarm, N=Normal).
2 – Title	The channel title, a maximum of 40 characters can be displayed.
3 – Fst	The digital state of the first sample (A=Alarm, N=Normal).
4 – Lst	The digital state of the last sample (A=Alarm, N=Normal).
5 – Fst-Change	The time the channel first changed state.
6 – Lst-Change	The time the channel last changed state.
7 – Changes	The number of times the channel-changed state.

Use the scroll bar or the up and down arrow keys to navigate through the digital channels and use the tab key to toggle between the analog and digital views.

Customizing the Analysis Display

The "Properties" option in the "File" menu allows for customizing the analysis display window. Below is a definition of each tab:

- **Analog Table** – The Analog Table tab is used to reorder, hide and show the columns in the Analog Table.
- **Analog Combination** – The Analog Combination tab is used to change the position of the data values in the Combination view.
- **Comtrade** – The Comtrade tab is used to define the Comtrade format for saving, the date and time format for display and for setting automatic conversion from RMS data to Peak data when using the "Save As Comtrade" feature.

- **Colors** – The Colors tab is used to define the background colors of each display section and to set the default analog channel colors.
- **Values File** – The Values File tab is used to define the Values File and general information used when saving samples values to a file.
- **Display Settings** – The Display Settings tab is used to define the Scale Multiplier for the traces and phasor/circular chart scaling. It also can define general display information for the window such as: setting the display trace type (sample based or time based), defining the Phase Angle Convention (Sine or Cosine), showing the time reference bar between the analog channels and the digital channels, showing the separator bar between multiple events displayed in one window, showing or hiding the fault bar (red dotted vertical bar), and defining the option to reference angles across windows when Sync mode is active. When "Yes" is defined for reference angles across windows all phase angles for the currently opened windows will be referenced from the first marked channel in the active window.
- **Append / Merge** – The Append/Merge tab is used to define from which file the samples will be discarded from when the discard common times option is used. It also is used to determine whether the station name should be added to the analog/digital titles when an append/merge option is executed.
- **Driver Data Type** – The Driver Date type tab is used to define the type of data that is saved to the displayed device's data file: RMS Type or Peak Type, convert the analog sample values between primary and secondary and for editing the ratio values.
- **Filters** – The Filters tab is used to define if spikes detected in the data file should be ignored when the maximum and minimum values are calculated and at what level they should be ignored.

Time & Sample Based Displays

The "Trace Display Type" field located in the "Display Settings" tab of the "Properties" dialog allows for toggling between the "Time Based" display and the "Sample Based" display. The sample based display plots the channel data with one-pixel distance between each displayed sample. Sample based displays are useful for showing changes in sampling frequency. The time-based display plots the channel data in time. Time based displays are useful for showing changes in line frequency.

To change the trace display type, open the "Properties" dialog under the "File" menu. Click the "Display Settings" tab and change the "Trace Display Type" field to time based or sample based.

Fault Reference Time Bar

The Fault Reference Time Bar is displayed between the analog and the digital traces. It displays the time difference from the fault time defined in the displayed file. The units are displayed in the Delta X status field.

To show or hide the fault reference time bar open the "Properties" dialog under the "File" menu. Click the "Display Settings" tab and select "Yes" or "No" for the "Show Reference Time Bar" field.

Superimposing Analog Channels

To superimpose two or more analog channels, mark the channels and press F7 or select the "Super Impose" menu option from the "Data" menu. The marked channels are superimposed and placed at the top of the display. If no channels are marked, all channels are superimposed. Press F7 to turn this feature ON/OFF.

Changing Analog Channel Colors

To change the color of an analog channel, click the opposite mouse button on the channel ID. Select the channel color from the list or click "More Colors" to select from the color palette. Channels must be unmarked to change their colors.

Mark Raw Values

Mark Raw Values marks the raw values saved in the active waveform file on disk. A small hollow blue circle is placed at the raw value. This feature is helpful in highlighting the raw sample values saved in low sampling rate files.

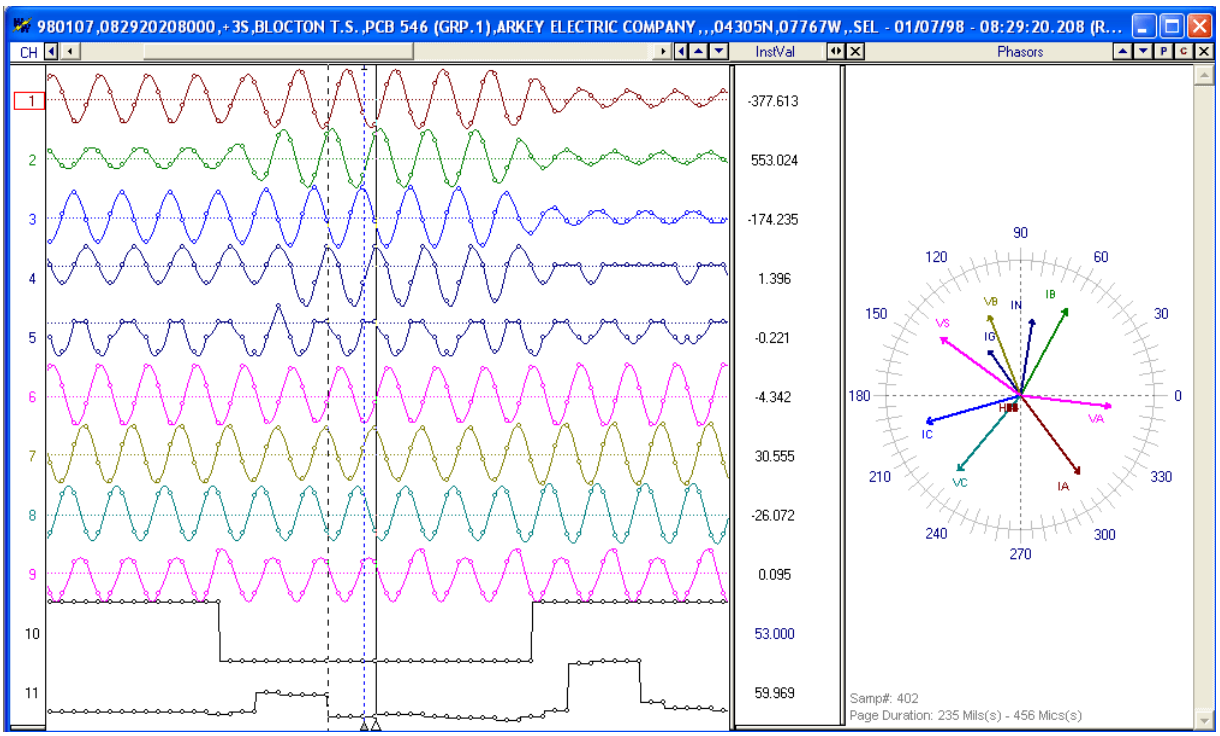


Figure 1.12 Mark Raw Values

Mark Peak Values

Mark Peak Values marks the peak sample values for all visible analog traces. A small solid gray square is placed at the peak values. This feature is helpful in highlighting the positive and negative peak values.

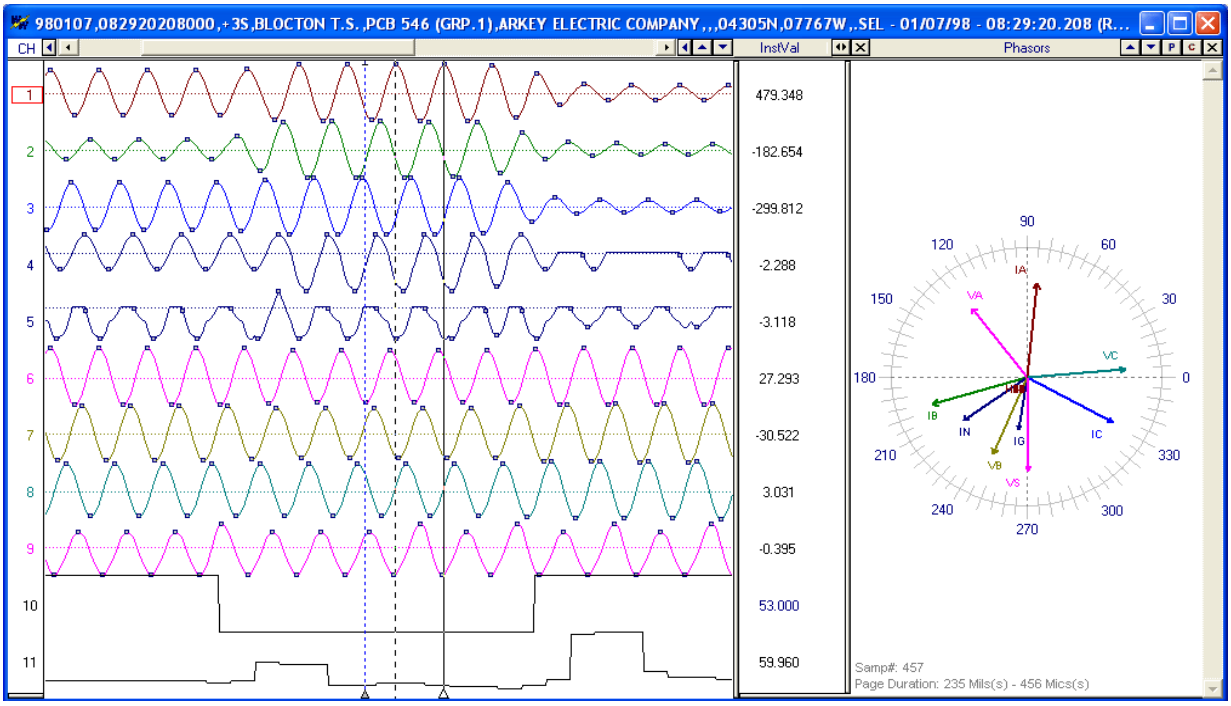


Figure 1.13 Mark Peak Values

Mark Change in Sign Values

Mark Change In Sign marks all samples where a change in sign occurs. A small solid gray triangle is placed at the change position. This feature is helpful in highlighting the position where a change in sign occurs in the signal.

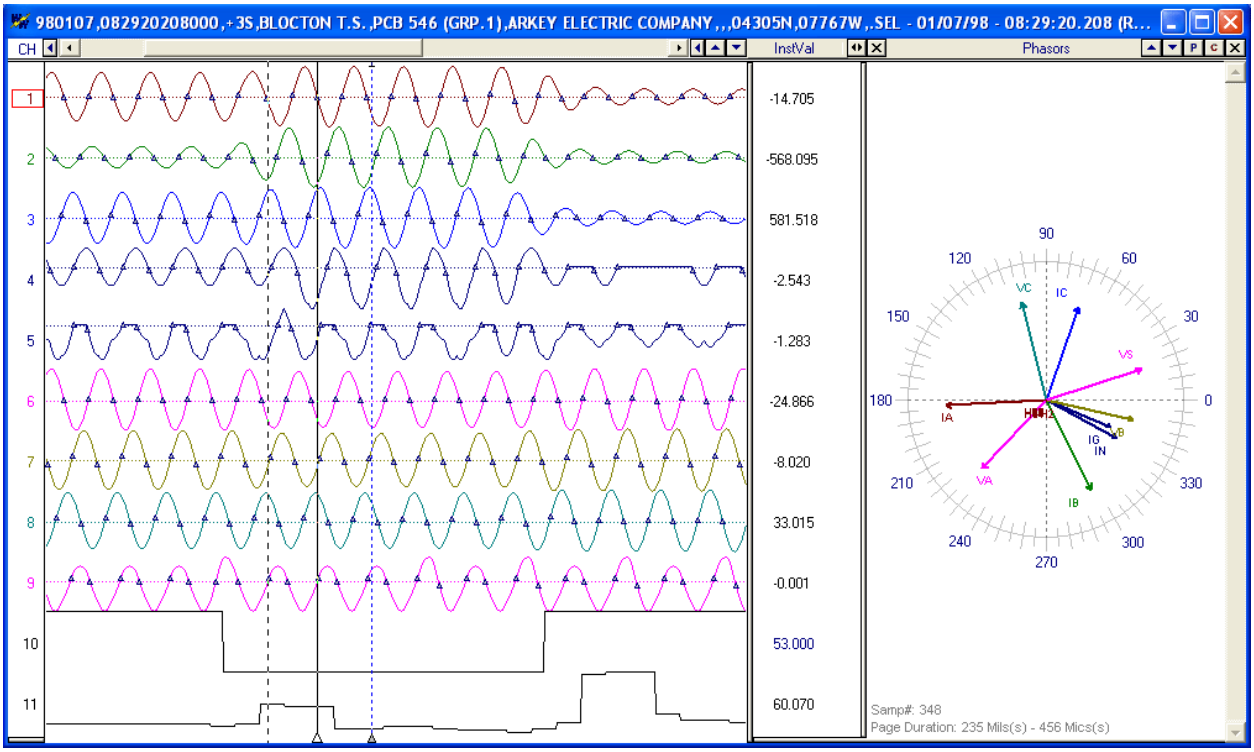


Figure 1.14 Mark Change in Sign

Change Analog Values (Primary ←→ Secondary)

The values displayed in the analog table are either in primary or secondary quantities. If the file defines the type of values saved the type is displayed in the status bar. Also, if the CT and PT ratios are defined in the configuration file the values can be changed from primary to secondary and vice versa. To change the values, open the properties dialog and click on the "Driver Data Type" tab, and select the Primary or Secondary radio button to switch between values.

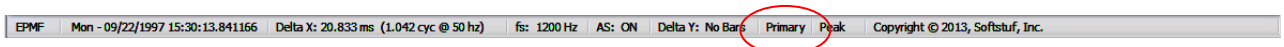
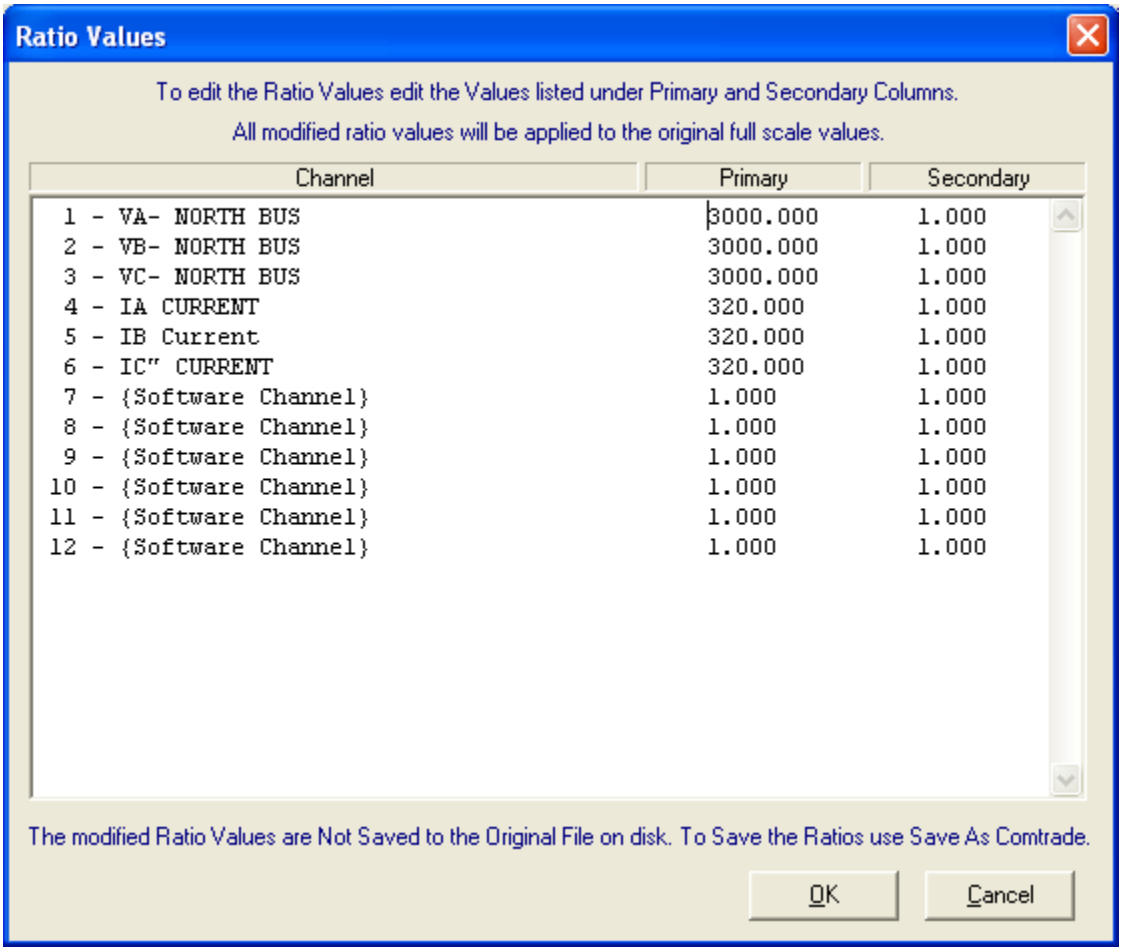


Figure 1.15 Type of Analog Values Displayed



Channel	Primary	Secondary
1 - VA- NORTH BUS	3000.000	1.000
2 - VB- NORTH BUS	3000.000	1.000
3 - VC- NORTH BUS	3000.000	1.000
4 - IA CURRENT	320.000	1.000
5 - IB Current	320.000	1.000
6 - IC" CURRENT	320.000	1.000
7 - {Software Channel}	1.000	1.000
8 - {Software Channel}	1.000	1.000
9 - {Software Channel}	1.000	1.000
10 - {Software Channel}	1.000	1.000
11 - {Software Channel}	1.000	1.000
12 - {Software Channel}	1.000	1.000

Figure 1.17 Ratio Values

Creating Virtual Channels

The analysis window allows for six software analog channels (SAC). These additional virtual channels exist only in RAM. The sample values are created using a function of the existing analog channels. Predefined operators can be used to calculate a missing phase, create positive/negative and zero sequence channels; convert channels to secondary or primary values; calculate V/I for fault resistance/impedance, multiply, divide, add and subtract multiple channels; multiply, divide, add and subtract channel data by a constant value; create an envelope of an analog channel; define over-trigger or under-trigger values; calculate a missing phase; define the prefix and unit for the channel.

All calculations are designed to operate "on the fly". For the forward-looking SAC operator ("@" some positive angle) care must be taken. Upon opening a file and while the system is reading the data samples, the forward samples are not available. In that case, the system uses the current sample instead of the requested forward sample. To execute forward looking SAC instructions, wait until the file is read and displayed.

SAC title and operators can be saved to an ASCII text file on disk by using the “Save” and “Save As” buttons located to the right of the SAC operators. The “Open” button allows for opening existing SAC files without having to manually enter the SAC titles and operators. These features are useful for reusing existing SAC operations on like files. The “New” button clears the existing SAC title and operators.

The SAC and SDC instructions are composed of an operator and an operand. An operand can be a channel defined by the channel number or a constant. Constant values must have a “^” operator before each value to distinguish

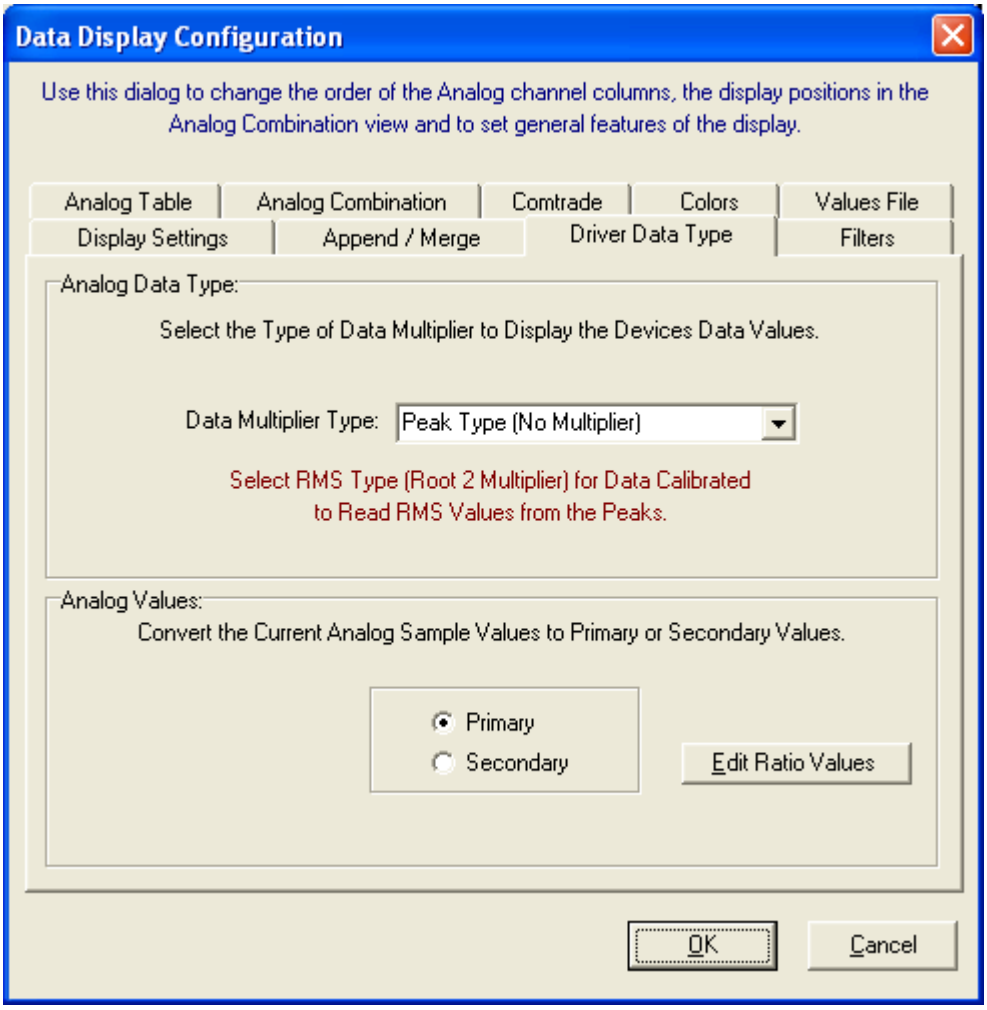


Figure 1.16 Change Analog Values (Primary ←→ Secondary)

The CT and PT ratio values can be edited by clicking on the “Edit Ratio Values” button in the “Driver Data Type” tab. The values are listed in a table format for each analog channel. The modified ratio values are not saved to the original file. To save the edited ratio values use the “Save As: Comtrade” option under the “File” menu.

between channel numbers and constant values. To phase shift analog channels, use the "@" sign before each angle defined. All angles must be defined in degrees. Following is a list of all the software operators that are available:

- "**+**" - Add (Analog),
- "**-**" - Subtract (Analog),
- "*****" - Multiply (Analog),
- "**:**" - Divide (Analog),
- "**^**" - Constant value (Analog),
- "**@**" - Phase Shift (Analog),
- "**e**" - Adjusted envelope (Analog),
- "**a**" - Envelope (Analog),
- "**<**" - Under-trigger (Analog),
- "**>**" - Over-trigger (Analog),
- **"h"** - Harmonic for Channel (Analog),
- "**h=**" - Harmonic for all Back Operations (Analog),
- "**x**" - real component (Analog),
- "**y**" -imaginary component (Analog),
- "**m**" -magnitude (Analog),
- "**d**" -angle (Analog),
- "**r**" -true RMS (Analog),
- "**f**" -cyclic frequency (Analog),
- "**q**" -instantaneous frequency (Analog),
- "**t**" -delta time frequency (Analog),
- "**s**" - sin operator (Analog),
- "**c**" - cos operator (Analog)
- "**b**" -operate between bars only (Analog),
- "**|**" - Absolute Value (Analog),
- "**p=**" - Prefix (Analog),
- "**u=**" - Unit (Analog),
- "**+**" - And (Digital),
- "**.**" - Or (Digital),
- "**/**" - Instruction terminator (Analog & Digital)

NOTE: All SAC operations are performed in Reverse Polish Notation (one operation at a time). The instruction set must always terminate with a "/". An operation error is generated if the instruction formats are not followed.

Operators are formatted as a stacked set of instructions. An instruction is composed of four attributes:

1. The operator: +, -, *, :, ...
2. The operand: channel index (1, 2, 3, ...) or constant value (such as ^3.14)
3. The function: @, h, x, y, m, d, f, ...
4. Instruction terminator: /

To display the SAC dialog, select "Software Analog Channels" from the Channels menu. Below are some examples:

Operations	Example	Description
Addition	+7/+8/+9/	Add channels 7, 8, & 9 and store the result in the SAC.
Subtraction	+7/-8/-9/	Subtract channel 8 from channel 7, and store the result in the SAC then subtract channel 9 from the SAC and restore the values in the SAC.
+ Sequence	+1/+2@120/+3@240/:^3/p=k/u=volt/	Calculate the + sequence components and store the result in the SAC then set the SAC's prefix and unit.
- Sequence	+1/+2@240/+3@120/:^3/p=k/u=volt/	Calculate the - sequence components and store the result in the SAC then set the SAC's prefix and unit.
0 Sequence	+1/+2/+3/:^3/p=k/u=volt/	Calculate the zero sequence components and store the result in the SAC then set the SAC's prefix and unit.
Harmonics	+16/h=1/p=k/u=volt/	Extract the 1st Harmonic component from Channel 16 and store in the SAC then set the SAC's prefix and unit.

Multiplication	+3/*^2.66/	Multiply all sample values in channel 3 with the constant value 2.66 and store the result in the SAC.
Division	+7/:3/	Divide all samples values in channel 7 by the sample values in channel 3 and store the result in the SAC.
Half Cycle Envelope	+2/e/	Calculate the half cycle envelope of channel 2 and store the result in the SAC.
Envelope	+12/a/	Calculate the envelope of channel 12 and store the result in the SAC.
Under-trigger	+4/<135/	Store all the sample values from channel 4 that are < 135 in the SAC.
Over-trigger	+62/>500/	Store all the sample values from channel 62 that are > 500 in the SAC.
Absolute Value	+7/+8/+9/|/p=k/u=Volts/	Add channels 7, 8, & 9 and store the absolute value of the result in the SAC then set the SAC's prefix and unit.
Frequency	+7f/u=Hz/	Store the cyclic frequency of channel 7, and set the SAC's unit to Hertz.
Frequency	+7q/u=Hz/	Store the instantaneous frequency of channel 7, and set the SAC's unit to Hertz.
Magnitude	+11m/u=V/p=k/	Store the magnitude of channel 11, and set the SAC's unit to Volt and the prefix to k.
Real	+4x/+5x/+6x/u=V/	Store the real components of the fundamental of 4, 5 and 6 and set the SAC's unit to Volts.
Imaginary	+4/+5/+6/y2/u=V/	Store the imaginary components of 2nd harmonic of 4, 5 and 6 and set the SAC's unit to Volts.

Secondary	+1/:^3000/u=V/	Store channel 1 values divide by 3000 and set the SAC's unit to V.
Impedance (V/I)	+4/:6/h=1/u=mho/	Store the fundamental of channel 6 (V) divided by channel 1 (I) and set the SAC's unit to mho.
Differential Current	+1/+2/+3/+4/|/u=A/	Store the absolute value of the sum of channels 1 2 3 and 4 and set the SAC's unit to A.
Apparent Power (1=V, 4=I)	*1m/*4m/p=k/u=Watts/	Store the magnitude of channel 1 multiplied by the magnitude of channel 4 and set the SAC's prefix to k and the units to Watts.
Power Factor	+1d/-4d/c/u=Deg/	Store the cosine of the angle of channel 1 minus the angle of channel 4 then set the SAC's unit to Deg.
Active Power	+1d/-4d/c/*1m/*4m/p=k/u=Watts/	Calculate the cosine of channel 1 angles minus channel 4 angles, then store the calculated value multiplied by channel 1 magnitudes and channel 4 magnitudes, set the SAC's prefix to k and the unit to Watts.
Reactive Power	+1d/-4d/s/*1m/*4m/p=k/u=Vars/	Calculate the sine of channel 1 angles minus channel 4 angles, then store the calculated value multiplied by channel 1 magnitudes and channel 4 magnitudes, set the SAC's prefix to k and the unit to Watts.

Multiplication	+3/*^2.66/	Multiply all sample values in channel 3 with the constant value 2.66 and store the result in the SAC.
Division	+7/:3/	Divide all samples values in channel 7 by the sample values in channel 3 and store the result in the SAC.
Half Cycle Envelope	+2/e/	Calculate the half cycle envelope of channel 2 and store the result in the SAC.
Envelope	+12/a/	Calculate the envelope of channel 12 and store the result in the SAC.
Under-trigger	+4/<135/	Store all the sample values from channel 4 that are < 135 in the SAC.
Over-trigger	+62/>500/	Store all the sample values from channel 62 that are > 500 in the SAC.
Absolute Value	+7/+8/+9/|/p=k/u=Volts/	Add channels 7, 8, & 9 and store the absolute value of the result in the SAC then set the SAC's prefix and unit.

Frequency	+7f/u=Hz/	Store the cyclic frequency of channel 7, and set the SAC's unit to Hertz.
Frequency	+7q/u=Hz/	Store the instantaneous frequency of channel 7, and set the SAC's unit to Hertz.
Magnitude	+11m/u=V/p=k/	Store the magnitude of channel 11, and set the SAC's unit to Volt and the prefix to k.
Real	+4x/+5x/+6x/u=V/	Store the real components of the fundamental of 4, 5 and 6 and set the SAC's unit to Volts.
Imaginary	+4/+5/+6/y2/u=V/	Store the imaginary components of 2nd harmonic of 4, 5 and 6 and set the SAC's unit to Volts.
Secondary	+1/:^3000/u=V/	Store channel 1 values divide by 3000 and set the SAC's unit to V.
Impedance (V/I)	+4/:6/h=1/u=mho/	Store the fundamental of channel 6 (V) divided by channel 1 (I) and set the SAC's unit to mho.
Differential Current	+1/+2/+3/+4/|/u=A/	Store the absolute value of the sum of channels 1 2 3 and 4 and set the SAC's unit to A.
Apparent Power (1=V, 4=I)	*1m/*4m/p=k/u=Watts/	Store the magnitude of channel 1 multiplied by the magnitude of channel 4 and set the SAC's prefix to k and the units to Watts.
Power Factor	+1d/-4d/c/u=Deg/	Store the cosine of the angle of channel 1 minus the angle of channel 4 then set the SAC's unit to Deg.
Active Power	+1d/-4d/c/*1m/*4m/p=k/u=Watts/	Calculate the cosine of channel 1 angles minus channel 4 angles, then store the calculated value multiplied by channel 1 magnitudes and channel 4 magnitudes, set the SAC's prefix to k and the unit to Watts.
Reactive Power	+1d/-4d/s/*1m/*4m/p=k/u=Vars/	Calculate the sine of channel 1 angles minus channel 4 angles, then store the calculated value multiplied by channel 1 magnitudes and channel 4 magnitudes, set the SAC's prefix to k and the unit to Watts.

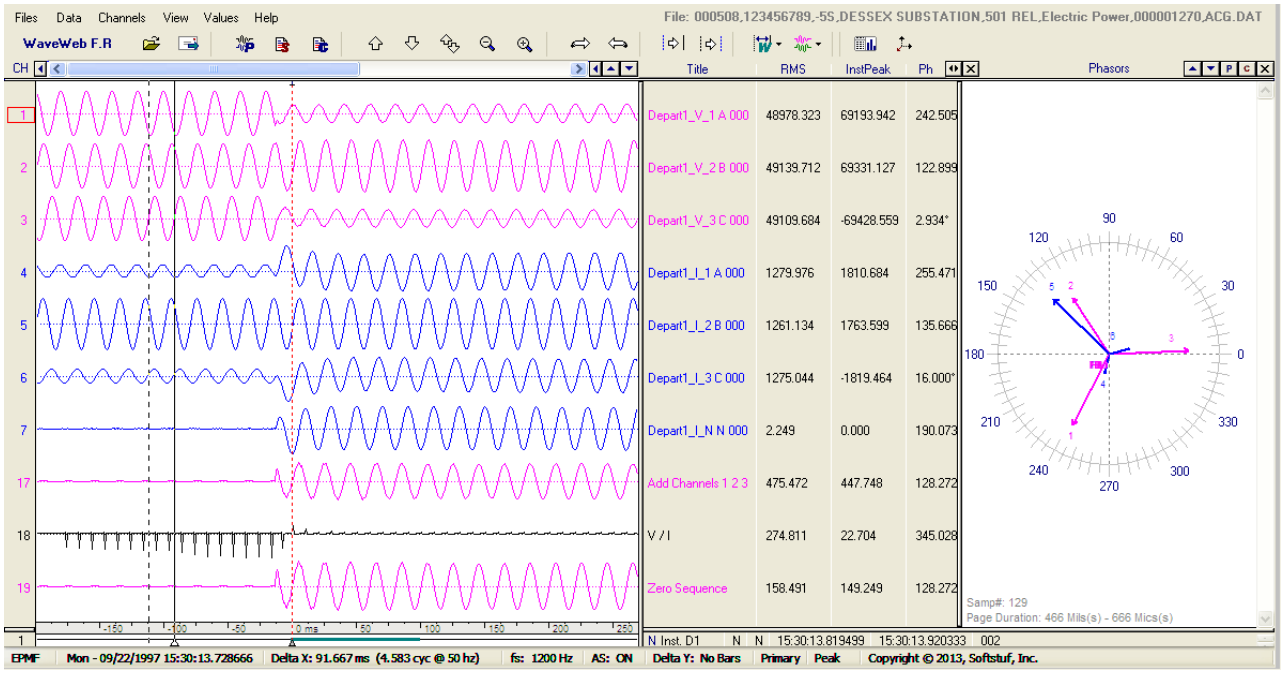


Figure 1.18 SAC Dialog

Engineers can use the additional channels as a generic tool for monitoring or modeling tasks. A virtual channel can be used to compute one of the phases of a monitored line by adding the remaining phases then subtracting the result from the residual channel. This in turn frees up a hardware channel for other monitoring needs.

The SAC operators and titles can be saved to an ASCII text file on disk to save time when re-entering SAC operator and titles. The SAC files can have any filename but the extension must be .SAC. If an extension is entered when saving a SAC file then the extension is deleted and .SAC is added to the filename. The active SAC path and filename is displayed in the first status field. The second status field indicates if the SAC title or operator fields where modified.

There are 4 options for the SAC files, Open, New, Save and Save As. Each option is explained below:

SAC File Operator	Description
Open	Open an existing SAC file. The Window's open file dialog is displayed. Navigate to the desired folder and double click on the SAC file. The SAC title and operator fields are populated with the contents of the selected file. If the file is not a valid SAC file then an error message is displayed.
New	Clear the current SAC title and operators and change the SAC filename in the first status field to Untitled. If the previous SAC title and operators where modified then a message will be prompted asking to save the existing SACs before clearing the fields.
Save	Save the active SAC title and operators to the SAC file listed in the first status field. If the SAC filename is listed as Untitled then the "Save As" dialog is displayed.
Save As	Save the existing SAC title and operators to a new SAC file. The Window's "Save As" dialog is displayed. Navigate to the desired folder and enter the new name in the "File name" field and click the "Save" button or press enter.

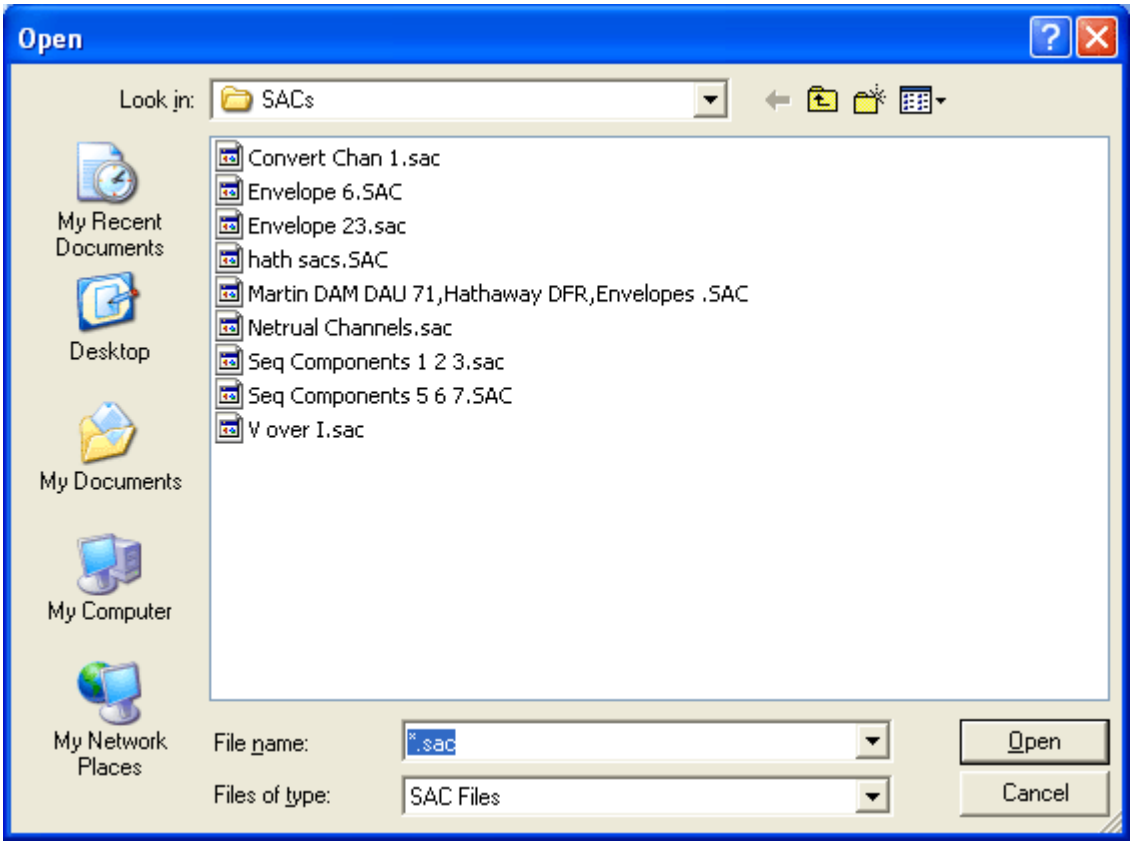


Figure 1.20 Open SAC File

Save As: CSV Format

The save as CSV format allows for saving specific analog information into a CSV comma delimited format. The "CSV Format" will save the RMS, Instantaneous or Vector (Magnitude and Angle or RMS and Angle) values to a comma delimited text file.

To save the analog channels in a CSV format select the Save As menu option under the “File” menu.

The first line in the CSV file is the header information for each channel. All the analog channels displayed in the active data plotting window are saved.

A dialog box is displayed to enter the destination path and the filename. The filename can be directly entered into the "File Name" field or the file can be automatically named using the IEEE C37.232 long file naming format. To have the file automatically named click on the "ComNames" check box. If the "ComNames" check box is checked then the File Name field will be disabled. To enter a file name, make sure the "ComNames" check box is unchecked.

The four Save As CSV options are:

- Save As CSV - RMS Values: Save the RMS Values in a comma delimited format.
- Save As CSV - Instantaneous Values: Save the Instantaneous Values in a comma delimited format.
- Save As CSV - Vector Values (Mag & Ang): Save the DFT Magnitude and Angle in a comma delimited format.
- Save As CSV - Vector Values (RMS & Ang): Save the RMS Value and Angle in a comma delimited format.

The file format saved is a comma delimited text file and the .CSV extension is automatically assigned. The first line in the file defines the header information. The first two columns are the samples date and time. Below is an example of the header:

DATE(0/DATE), TIME(1/TIME), IA(4/Amps), IB(4/Amps), IC(4/Amps), VA(4/Volts), VB(4/Volts),VC(4/Volts)

System Keys

This section lists the function keys, cursor keys, and menu buttons available in the device manager, query fields and DXF display.

Analysis

Cursor Keys	Description
Left Arrow	Move the data bar to the left one sample.
Right Arrow	Move the data bar to the right on sample.
Ctrl+Right Arrow	Move the data bar to the next peak for the first display channel or the first marked channel.
Ctrl+Left Arrow	Move the data bar to the previous peak for the first display channel or the first marked channel.
Shift+Ctrl+Right Arrow	Move the data bar ahead one cycle for the first displayed channel or the first marked channel.
Shift+Ctrl+Left Arrow	Move the data bar back one cycle for the first displayed channel or the first marked channel.
Shift+Left Arrow	Shift the analog information table to the left by one column.
Shift+Right Arrow	Shift the analog information table to the right by one column.
Page Up	Page up through the data.
Page Down	Page down through the data.
Home	Move the data bar to the first data sample.
End	Move the data bar to the last data sample.
Ctrl+Up Arrow	Increase the amplitude for all or marked channels.
Ctrl+Down Arrow	Decrease the amplitude for all or marked channels.
Ctrl+Page Up	Expand the time scale for all visible channels.
Ctrl+Page Down	Condense the time scale for all visible channels.
Tab	Toggle between the analog and digital channels.
Up Arrow	Move the cursor up one channel.
Down Arrow	Move the cursor down one channel.
Shift+Page Up	Display the analog/digital channels on the previous page.
Shift+Page Down	Display the analog/digital channels on the next page.
Ctrl+Home	Display the first page of the analog/digital channels.
Ctrl+End	Display the last page of the analog/digital channels.
Spacebar	Mark or Unmark the channel at the cursor position.
Shift+Up Arrow	Mark or Unmark a group of channels while moving the cursor up.
Shift+Down Arrow	Mark or Unmark a group of channels while moving the cursor down.
Insert	Display the hidden channels that were removed by the delete keys.
Delete	Hide the marked channels and respace the unmarked channels.
Enter	Hide the unmarked channels and respace the marked channels.

Esc	Display all the hidden channels or exit the data window.
Backspace	Display all the hidden channels.
+	Shift all the marked channels up one position.
-	Shift all the marked channels down one position.
Ctrl-A	Move the reference bar to the sample at the cursor bar.
Ctrl-Z	Move the RMS bar to the sample at the reference bar.

Menu Buttons	Description
Open File	Open a new waveform file.
Email Active File	Email the active file and any support files needed.
Properties	Display the Window Properties dialog.
Summary	View the Analog/Digital Summary of the active displayed file.
Recorder Chans	Display the waveform's analog/digital channel headers and scale factors.
Inc	Magnify the amplitude of the marked channels.
Dec	Attenuate the amplitude of the marked channels.
ASV	Turn auto scaling ON/OFF for all visible channels.
In	Condense the time scale of the visible channels.
Out	Expand the time scale of the visible channels.
View Marked	Hide the unmarked channels and respace the marked channels.
View All	Replot all the hidden channels.
Set Ref Bar	Move the reference bar to the sample at the cursor bar.
SetRMS Bar	Move the RMS bar to the sample at the reference bar.
Resize Sliding Window	Resize the RMS sliding window.
Channel Color	Change the Analog channels color at the cursor position.
Harmonics	Display the Harmonics table / histogram window.
Vector Harmonics	Show / Hide the vector harmonics in the phasor display.

APPENDIX G

Basic SMTP Email Control

Method

Description	SMTPSendEmail *ServerName, UserName, UserPassword, FromAddr, Subject, Body, Recipients, Optional Attachment, Optional Attachments* method to send basic SMTP email
Syntax	Private Sub Object1_Click( ) Object.SMTPSendEmail *ServerName, UserName, UserPassword, FromAddr, Subject, Body, Recipients, Attachment, Attachments* End Sub
Parameter(s)	ServerName (String): Specifies email server UserName (String): Specifies email user UserPassword (String): Specifies email password FromAddr (String): Specifies email user from address Subject (String): Specifies email subject Body (String): Specifies email body content Recipients (String): Specifies email recipients - [;] separator Attachment (String): Specifies a single email attachment (file path) - Optional Attachments (String): Specifies a folder where all files would be attached to the email - Optional

Extended SMTP EmailX Control

Methods

Description	Email_Initialize method to initializes the email (must be executed first)
Syntax	Private Sub Object1_Click( ) Object.Email_Initialize End Sub
Parameter(s)	None

Description	SMTPSendEmail method to send SMTP email
Syntax	Private Sub Object1_Click( ) Object.SMTPSendEmail End Sub
Parameter(s)	None

Properties

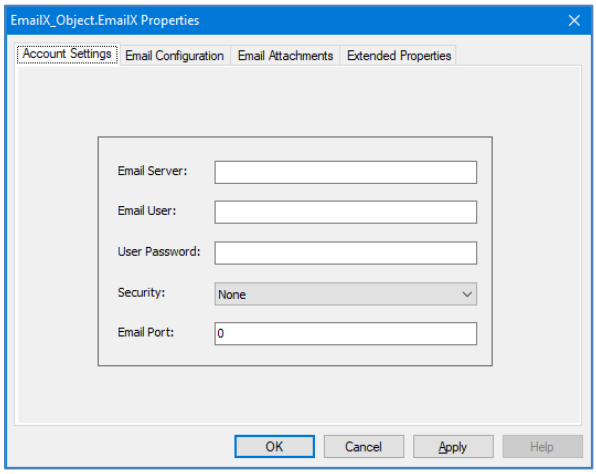


Figure #AG1

Property	Description	Example
ServerName	Specifies email server (string)	Object.ServerName = “smtp.mail.yahoo.com”
UserName	Specifies email user email address (string)	Object.UserName = “user@yahoo.com”
UserPassword	Specifies email user password (string)	Object. UserPassword = “ABC”
Security	Specifies if security is used (integer)	Object.Security = 1000 ‘- None Object.Security = 0 ‘- Direct SSL/TLS Object.Security = 1 ‘- Start TLS
Email_Port	Specifies email port (integer)	Object. Email_Port = 465

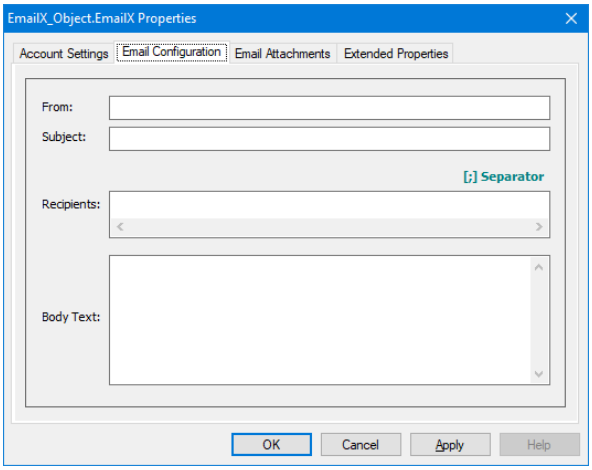


Figure #AG2

Property	Description	Example
FromAddress	Specifies from what user email was send (string)	Object.FromAddress = “SCADA1@Alarm1”
Subject	Specifies email subject text (string)	Object.Subject = “My Subject”
Recipients	Specifies email recipients (string)	Object.Recipients = “user@yahoo.com” & “; ” & “user@yahoo.com”
BodyText	Specifies email body text (string)	Object.BodyText = “My Body Text”

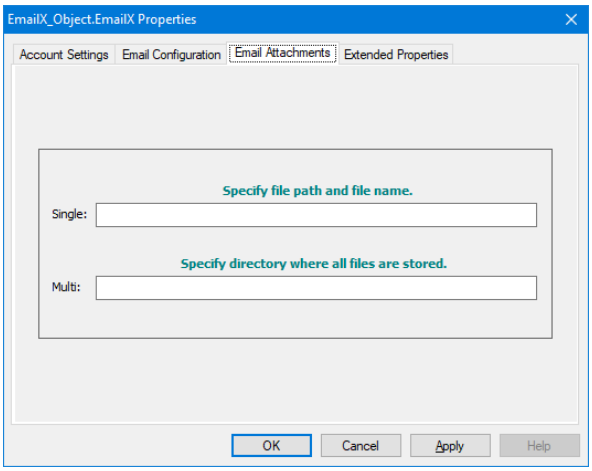


Figure #AG3

Property	Description	Example
Attachment	Specifies email attachment (string)	Object. Attachment = “C:\Tmp\myfile.txt”
Attachments	Specifies email folder (string)	Object. Attachments = “C:\Tmp”