Integration Objects'

Solution for OPC A&E Controls

OPC A&E ActiveX

Version 2.0Rev.2

User Guide

Compatibility

OPC A&E 1.02
OPC A&E 1.10

Integration Objects' OPC AE ActiveX User's Guide Version 2.0Rev.2
Published May 2018

TABLE OF CONTENTS

TABLE OF FIGURES

LIST OF TABLES

PREFACE

About this Guide

This guide:

- Explains how to install, configure and use the Integration Objects' OPC AE ActiveX, also referred to as OPC AE Controls.
- Describes the system requirements for this release.

Audience

This manual is intended for programmers who wish to use Integration Objects' OPC AE ActiveX. It assumes that readers are familiar with Microsoft ActiveX technologies and have experience with the programming language (for example, Visual Basic) needed to create their ActiveX application.

It also assumes that you have an idea about OPC (OLE for Process Control) and OPC Alarms and Events specification.

Related Documentation

OPC Foundation
As you use this user guide, you may also find the following specification useful: A&E Custom Interface Standard Version 1.10.

ActiveX Documentation
For more information about using ActiveX, refer to your ActiveX container documentation (for example, the Visual Basic documentation) or see Microsoft's web page: http://www.microsoft.com

CUSTOMER SUPPORT SERVICES

Phone	Email
Americas: +1 713 609 9208 **Europe-Africa-Middle East** +216 71 195 360	Support: customerservice@integrationobjects.com Sales: sales@integrationobjects.com Online: www.integrationobjects.com

GETTING STARTED

This chapter describes the ActiveX Control, the basic steps for using it, and installation information.

1. Overview

ActiveX control is one element of the Microsoft's ActiveX object-based technology. It is a piece of software that can be used with any application or programming environment defined as an ActiveX container. The container provides a framework in which the control operates, and collectively, with other ActiveX components, container and control provide whatever functions are required of the ActiveX software design. This requirement may be to interact with other ActiveX controls or containers or to run as a complete application.

Integration Objects' OPC AE ActiveX provides access to the basic features of the OPC AE API. Using these controls, you can develop an AE application that allows users to view real time alarms received by an OPC AE Server.

If you are a developer creating OPC client applications, this toolkit was designed for you. This customizable control can be embedded into any Web page or Windows® application.

The control can be used with any ActiveX container, including applications such as Microsoft Access or programming environments such as Visual Basic.

2. Software Components

The software contains the following software components and documents:

- ***Alarm Logger Control***: this control allows users to display current alarms and events in a list view.
- ***Alarm Logger .Net Control***: this control allows users to display current alarms and events in a list view. This is the preferred logger control to be deployed with .Net based ActiveX containers.
- ***AE Servers ListBox***: This control offers the possibility to retrieve OPC AE Servers in a list box.
- ***AE Servers ComboBox***: the same as the previous control, the AE Servers ComboBox is a combo box control, which retrieves the OPC AE Servers registered locally or remotely.

- ***Event Client Component***: This component allows VB and OLE containers to quickly and easily access data from any OPC A&E Server. Methods and properties exist within the control to allow the developed application to connect to the AE Server on the local or on the remote machine and then to access OPC AE Server interfaces.
- ***Event Client .Net Component***: This component allows VB and OLE containers to quickly and easily access data from any OPC A&E Server. Methods and properties exist within the control to allow the developed application to connect to the AE Server on the local or on the remote machine and then to access OPC AE Server interfaces. This is the preferred event client control to be deployed with .Net based ActiveX containers.
- Sample application with source code is provided to demonstrate the capabilities of the toolkit.
- The software User Guide (this document).
- The software Quick User Guide.

3. Features

This section contains a brief description of each of the components of Integration Objects' OPC AE ActiveX software.

Integration Objects' OPC Alarm Logger

- The ***Alarm Logger Control*** delivers a great GUI enhancement to VB applications - in other words, the ability to dock or float regular VB controls that connect and display data without any line of the code.
- Auto-discovery of all OPC AE servers available on the network.
- Connection to an AE OPC Server locally or remotely.
- Alarm filtering to view only certain types of incoming alarms.
- By adding customized filter constraints for incoming alarms, users can simply define filters using wildcards such as (?, *).
- Acknowledge events operator.
- Special text and background colors for alarms.
- Notifications of particular alarms sent to administrator by email.
- Archiving alarms automatically in CSV files.
- The user can follow the operations handled by the control by saving log entries as files.
- Enabling OLE Automation containers (such as VB, VBA, Internet Explorer) to view incoming alarms.
- Complete scrollable display.
- Change column information to be viewed.
- Configurable message filters to view only certain types of messages.
- Display server information on demand.

Integration Objects' OPC AE Alarm Logger .Net ActiveX

- The ***Alarm Logger .Net Control*** delivers a great GUI enhancement to VB applications - in other words, the ability to dock or float regular VB controls that connect and display data without any line of the code.
- Auto-discovery of all OPC AE servers available on the network.
- Connection to an AE OPC Server locally or remotely.
- Alarm filtering to view only certain types of incoming alarms.
- By adding customized filter constraints for incoming alarms, users can simply define filters using wildcards such as (?, *).
- Acknowledge events operator.
- Special text and background colors for alarms.
- Notifications of particular alarms sent to administrator by email.
- Archiving alarms automatically in CSV files.
- The user can follow the operations handled by the control by saving log entries as files.
- Enabling OLE Automation containers (such as Power Builder, Proficy iFIX, Proficy Cimplicity) to view incoming alarms.
- Complete scrollable display.
- Change column information to be viewed.
- Configurable message filters to view only certain types of messages.
- Display server information on demand.
- Print the incoming alarms

Integration Objects' OPC Event Client .Net ActiveX

- Auto-discovery of all OPC AE servers available on the network in a ListBox format.
- Connection to OPC AE servers locally and remotely
- Enabling OLE Automation containers (such as VB, VBA, Internet Explorer) to view incoming alarms.
- Acknowledge events operator.
- Display server information on demand.
- Alarm filtering to view only certain types of incoming alarms.
- Complete scrollable display.

Integration Objects' OPC Event Client ActiveX

- Auto-discovery of all OPC AE servers available on the network in a ListBox format.
- Connection to OPC AE servers locally and remotely
- Enabling OLE Automation containers (such as Power Builder, Proficy iFIX, Proficy Cimplicity) to view incoming alarms.
- Acknowledge events operator.
- Display server information on demand.
- Alarm filtering to view only certain types of incoming alarms.
- Complete scrollable display.

Integration Objects' Event Servers ListBox

- Auto-discovery of all OPC AE servers available on the network in a ListBox format.

Integration Objects' Event Servers ComboBox

- Auto-discovery of all OPC AE servers available on the network in a ComboBox format.

4. Software Requirements

Microsoft .Net Framework version 2 Redistributable Package or a more recent version is required for a proper installation of the OPC AE ActiveX.

To develop applications using Integration Objects' OPC AE ActiveX, you will require the following:

- Operating System :
  - Windows XP
  - Windows Server 2003
  - Windows 7
  - Windows 8 and 8.1
  - Windows 10
  - Windows Server 2008
  - Windows Server 2012
  - Windows Server 2016

- ActiveX/COM aware programming environment, such as Visual Basic®, Delphi™, etc. (only on the development platform).

Microsoft Visual VB 6.0 (with Service Pack 4) or higher may be required as well, for users to develop their own applications.

5. OPC Compatibility

- OPC A&E specification version 1.02
- OPC A&E specification version 1.10

6. Installing OPC AE ActiveX

The product package consists of several components that need to be loaded in your system. The setup application is designed to place all necessary files into the correct locations.
Follow the installation wizard instructions. Click *Next* to proceed from screen to screen in the installation program. You can use the *Back* button to return to earlier screens and change any information that is incorrect. Click *Cancel* to cancel the installation process at any time.

The following section includes the screen shots taken during the installation process.

1. Double-click the *Setup.exe* and the Welcome to the Integration Objects' OPC AE ActiveX Setup Wizard window will appear (see Figure 1).

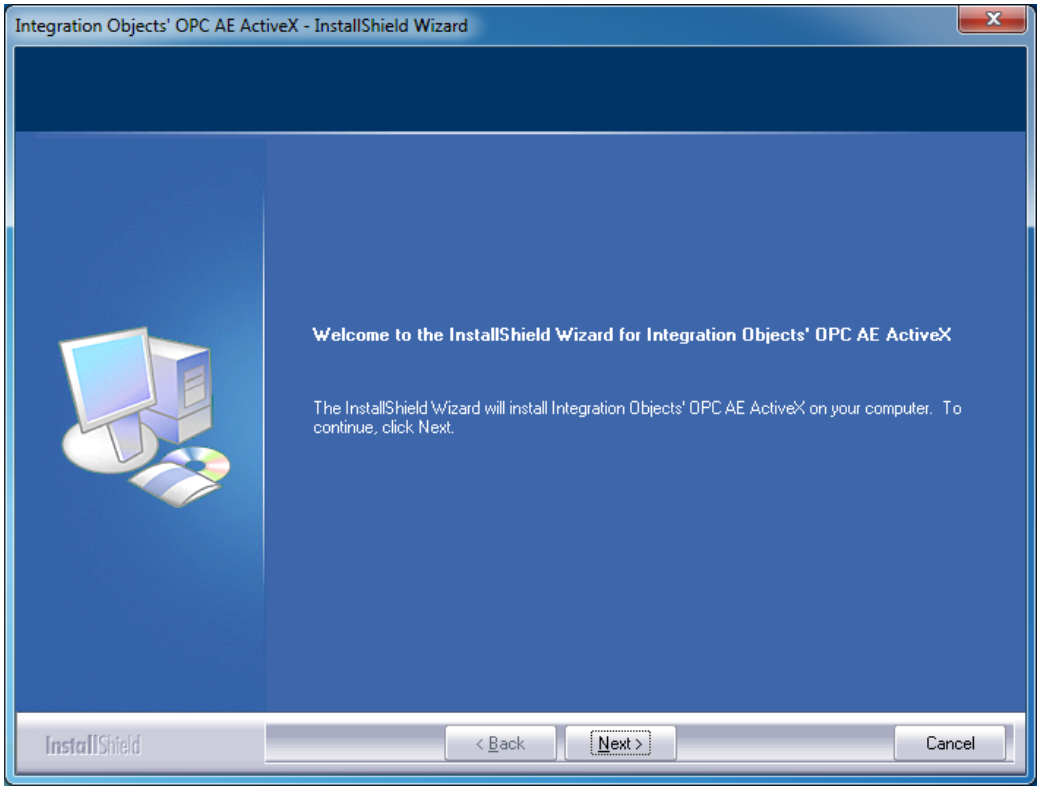


Figure 1 : Welcome Setup Wizard

2. Click the *Next* button to begin the installation process.

3. After the welcome window is dismissed, the license window shown below in Figure 2 is displayed. Please read the license agreement and select “I Agree” to proceed with the installation.

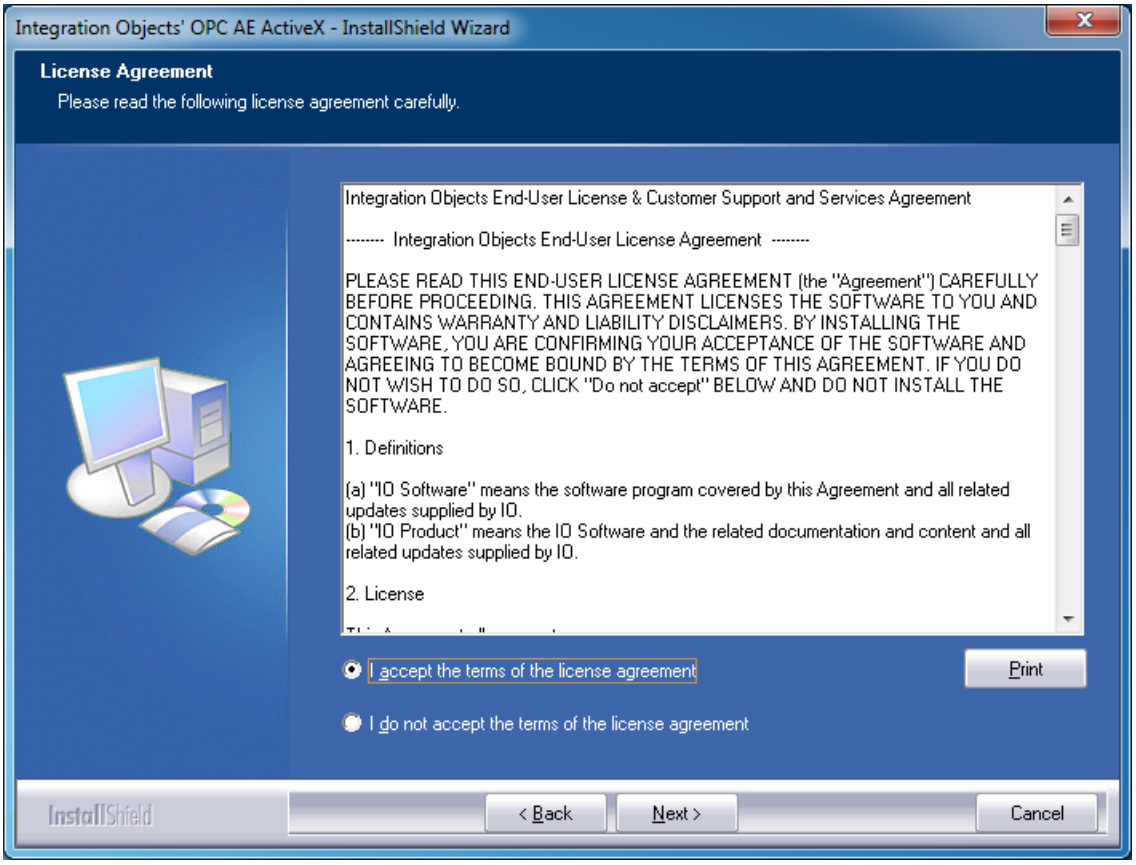


Figure 2: License Agreement Window

4. After you agree to the software license, the window shown in Figure 3 is displayed. This window asks you to introduce information about yourself and your company.

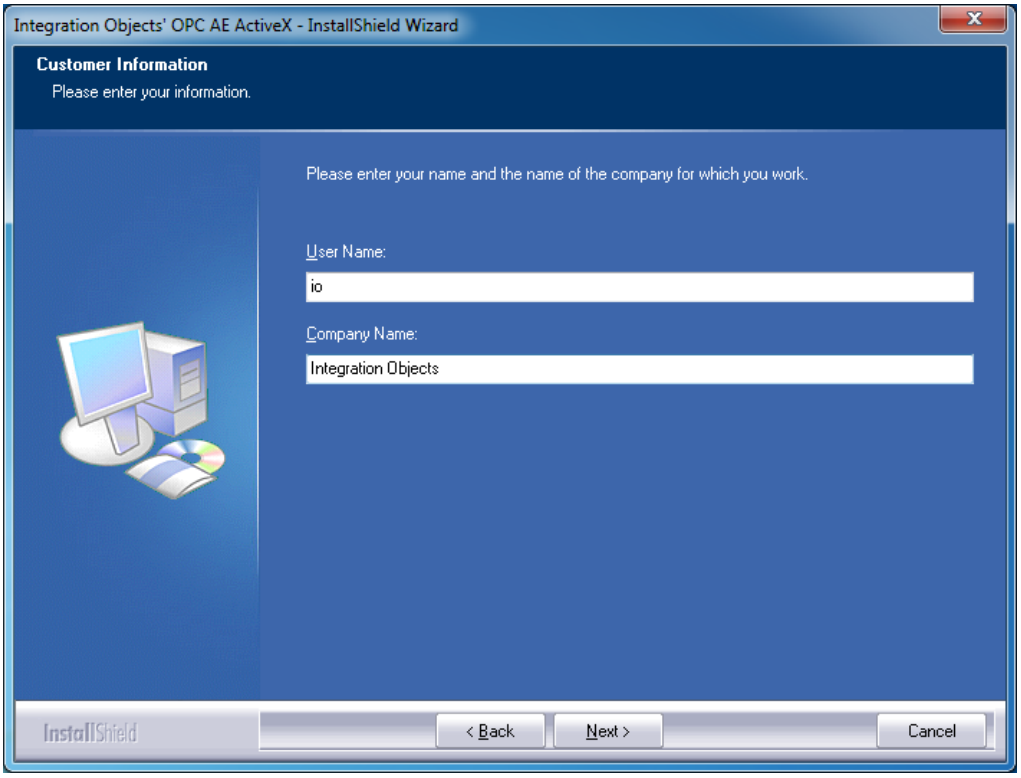


Figure 3: Customer Information

5. A folder selection window appears when you click the next button (see Figure 4).
6. By default, all components will be installed under the directory "\Program Files\Integration Objects\Integration Objects' OPC AE ActiveX" unless the user specifies another location during the installation. In most instances, the default folder is the best place for installing the software. If a different directory must be used, click the *Browse* button to search for the directory where you want to install the product and then click the *OK* button.

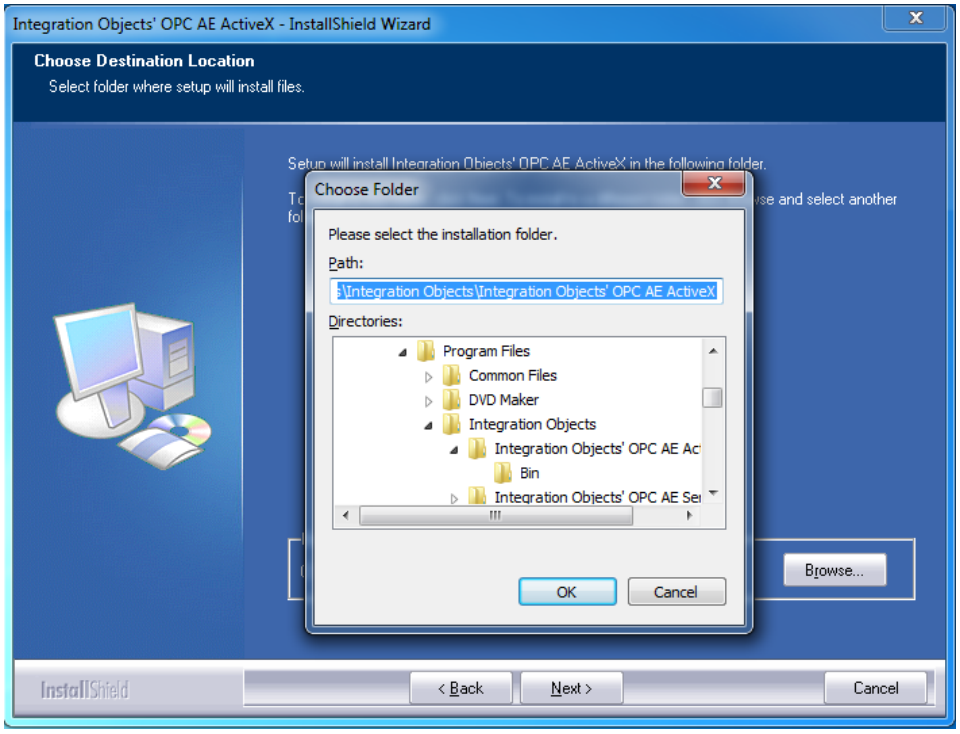


Figure 4: Select Destination Folder

7. Click *Next* and a confirm installation window appears (see Figure 5).

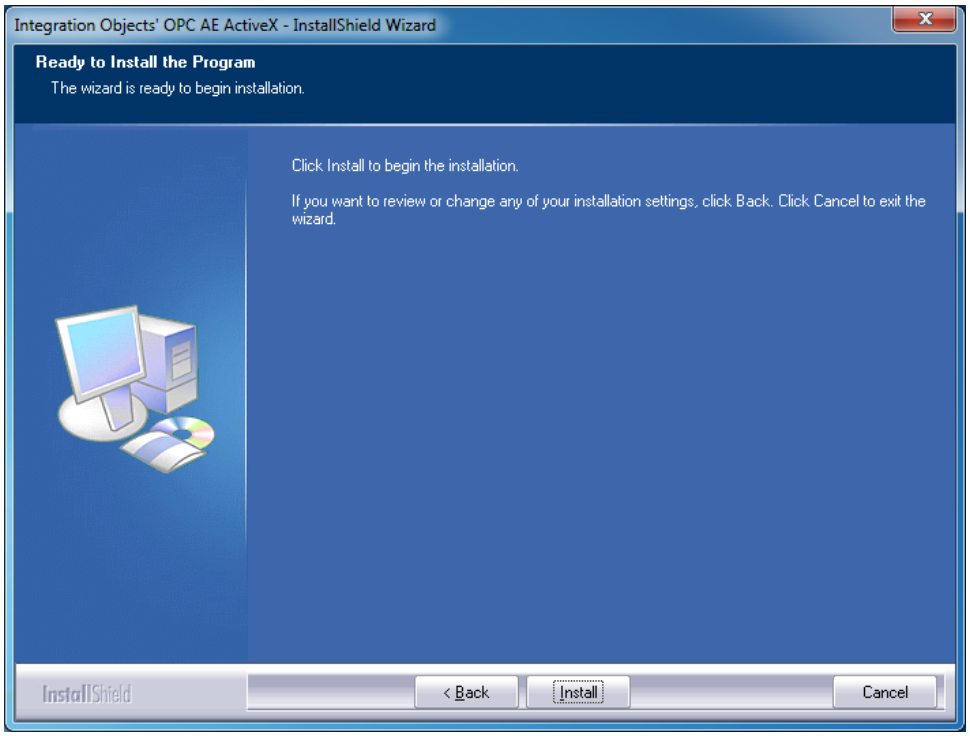


Figure 5: Confirm Install

8. To begin installation, click the *Next* button. An installation progress window will appear (see Figure 6).

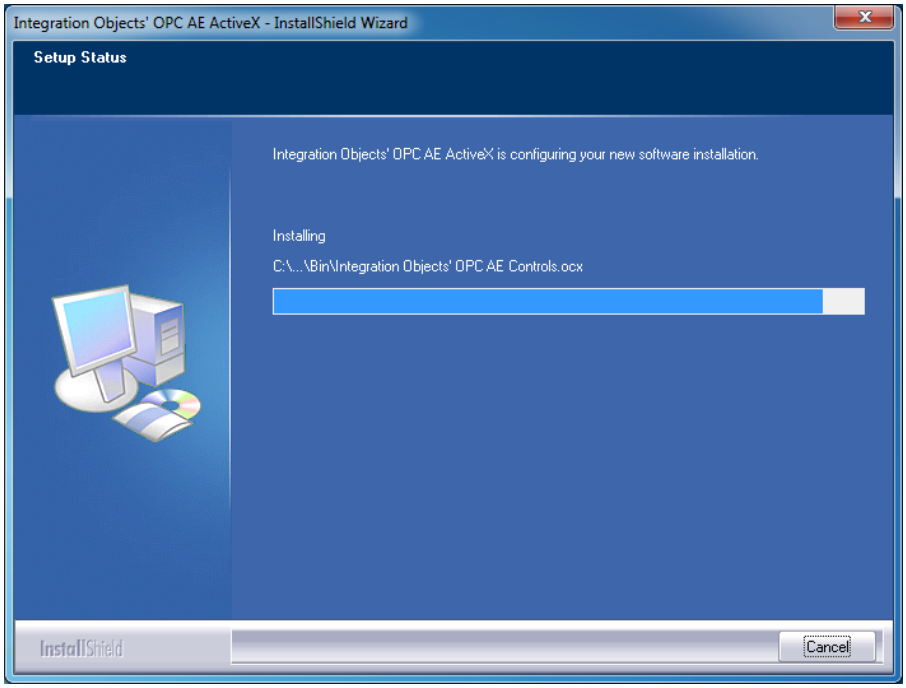


Figure 6: Installation Progress Window

9. An Installation Complete window will appear when the product has been successfully installed (see Figure 7).

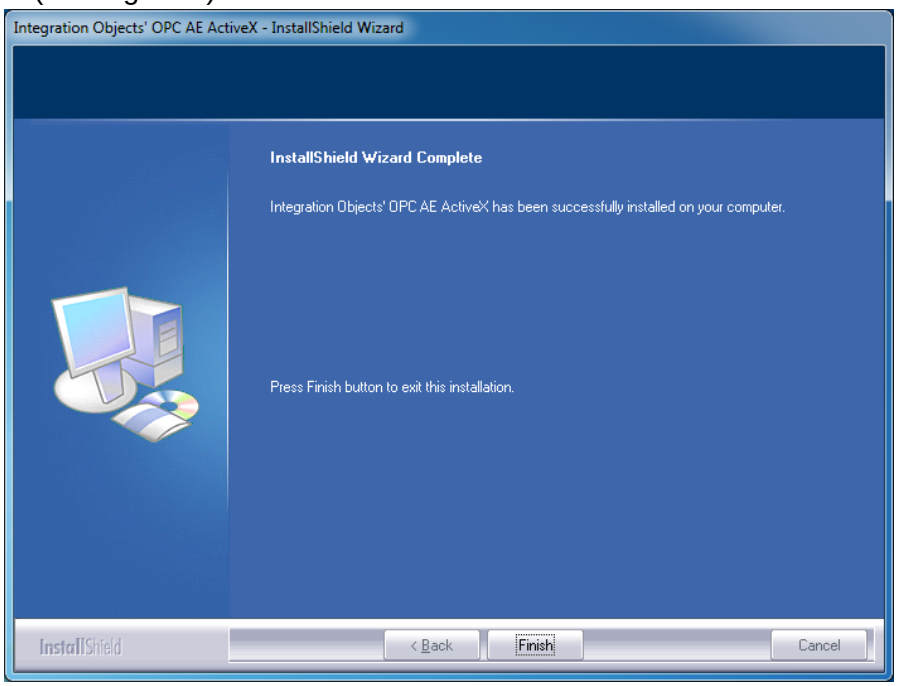


Figure 7: Installation Complete Window

10. Click the *Finish* button to exit the installation window.

The installation copies all necessary files to the target computer, creates a short-cut icon to place the configuration tool in the Start menu and makes an un-installation entry in the Add/Remove Programs Window in the Control Panel.

Click on Start ➔ Programs ➔ Integration Objects ➔ OPC AE ActiveX

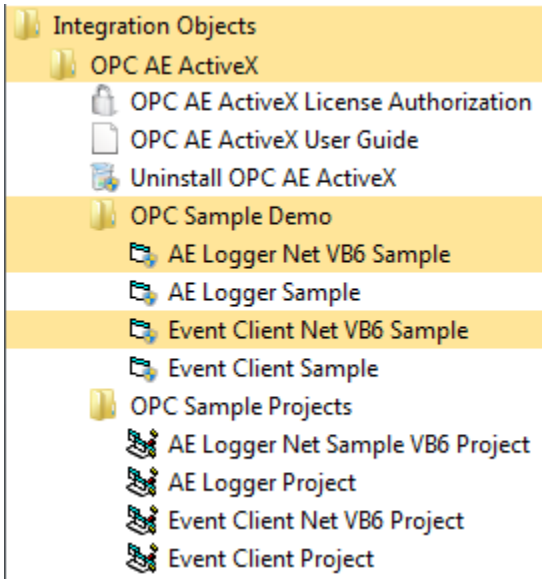


Figure 8: Start Menu

7. Uninstalling OPC AE ActiveX

First you should close all applications using the control. Then you uninstall it by using the **Add/Remove Programs** wizard from MS Windows **Control Panel**.

To uninstall the toolkit:

1. Make sure there are no running applications using Integration Objects' OPC AE ActiveX.
2. Open the Add/Remove Programs control panel.
3. Select "Integration Objects' OPC AE ActiveX", and click Change/Remove. The uninstall process completely removes all files that were installed. This does not remove files and folders that were created after the installation.
4. After you run the uninstall program, check the installation directory and remove any unnecessary files and sub directories. After uninstalling, there may still be files stored in the installation directory. You should delete these as well.

The software can also be removed using the ""**Uninstall OPC AE ActiveX**" shortcut available in the start menu.

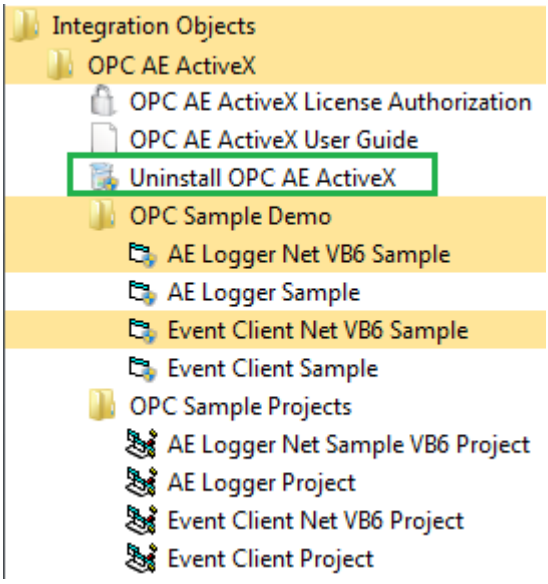


Figure 9: Start Menu – Uninstaller Shortcut

If you are using Windows 10, Windows Server 2012 or Windows Server 2016 operating systems, the uninstaller needs to be run from the start menu as illustrated below.

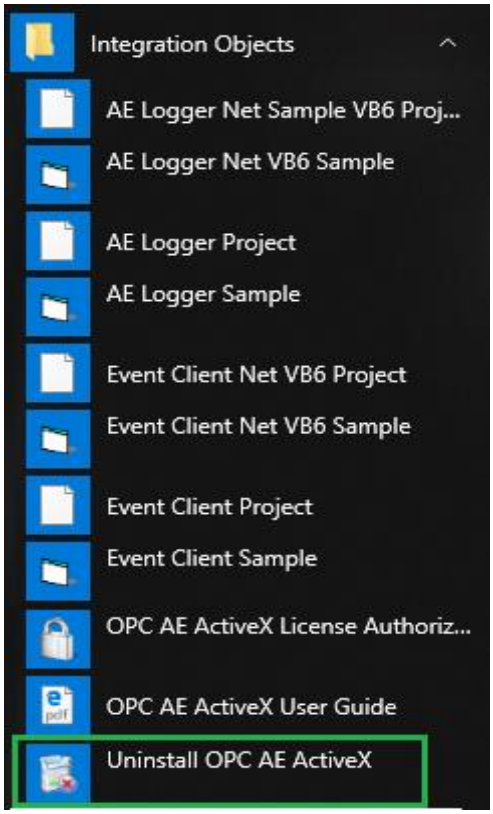


Figure 10: Windows 10 Startup Menu - Uninstall Shortcut

Installed Files

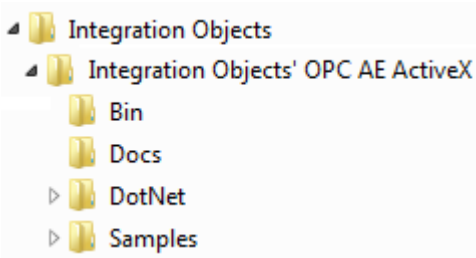


Figure 11: Integration Objects' OPC AE ActiveX Installation Directory

When the installation is complete, you should see the following directories installed under the destination folder that you selected:

Directory	Definition
Bin	This directory contains the needed ocx, dll files.
Docs	This directory contains this user guide (.pdf).
DotNet	This directory contains the .Net ActiveX controls dll files
Samples	This directory contains the VB6 samples included within the product.

Table 1: Installed Directories

8. Dependencies

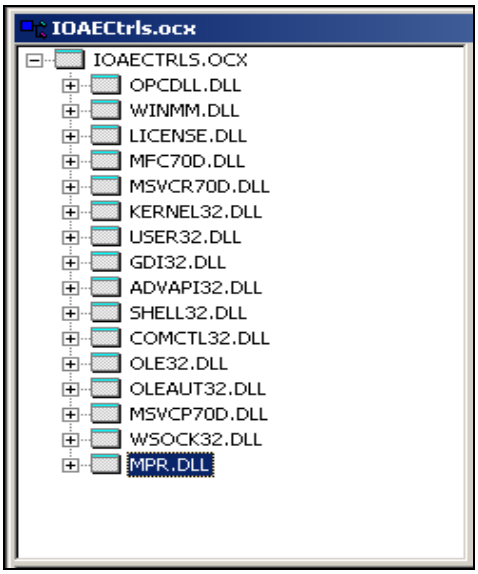


Figure 12: Dependencies

Integration Objects' OPC AE ActiveX requires the Microsoft Foundation Class (MFC) Runtime modules. These modules are usually found in the Windows system directory.

Integration Objects' OPC AE ActiveX dependencies include the following files: mfc70D.dll, msvcp70.dll, msvcr70.dll, mpr.dll, opcdll.dll, WSOCK32.dll.

There may be additional files provided by Windows which are not listed here. If the control fails to register with the above files, you may use the Dependency Walker installed with the Visual Studio to get the dependencies list.

USING OPC AE ACTIVEX

This chapter will explain how to program user applications. A simple application is provided with source code to help users get started using Integration Objects' OPC AE ActiveX. The sample source code will be stored in the "Samples" folder under the product installation location.

You can insert the product ActiveX controls in:

- A VB window (or a window built with another application development tool that supports ActiveX).
- An HTML page.

The first part of this chapter focuses on the steps needed to develop a minimal OPC application using Integration Objects' OPC AE ActiveX with Microsoft Visual Basic 6.0.

1. Start Example

When starting to write an application, first add the component to your VB project. The following steps show how to incorporate the control into a VB6.0 project.

Step 1: Start Visual Basic. Choose *New Project* | *Standard EXE*. A project named Project1 with a form called Form1 will be displayed.

Step 2: Select *Project* | *Components*. Uncheck "Selected Items Only" box to show all components, and check two components called ***Integration Objects' OPC AE Controls Control module*** and ***Integration Objects' OPC Event Client ActiveX***. If these components are not listed here this means that the product was not correctly installed. Refer to the Install section about how to install and register these components for design and development purposes.

Step 3: Once these controls are available in your project, new icons on the left toolbar will appear, and the user can simply drag and drop any of these icons to the Form. Four controls are available: ***Integration Objects' OPC Alarms Logger***, ***Integration Objects' Event Servers ListBox***, ***Integration Objects' Event Servers ComboBox*** and ***Integration Objects' OPC Event Client ActiveX.***

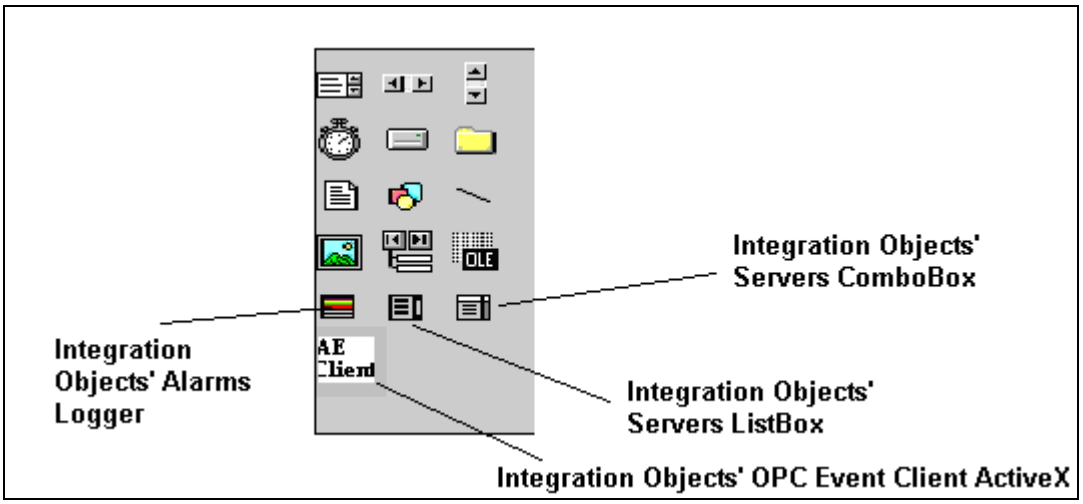
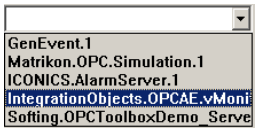
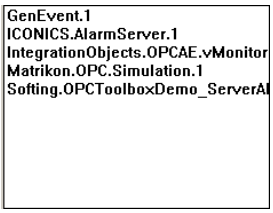


Figure 13: Integration Objects' OPC AE ActiveX

Integration Objects' Servers ListBox and ***Integration Objects' Servers ComboBox*** are simple controls which allow users to get locally or remotely registered AE Servers into ListBox or ComboBox.





Integration Objects' Alarms Logger allows the user to view current alarms and handles alarm acknowledgement. Below is a VB Form including ***Integration Objects' Alarms Logger.*** This control represents a list view. Users can modify the layout of the information, including order, and displayed data.

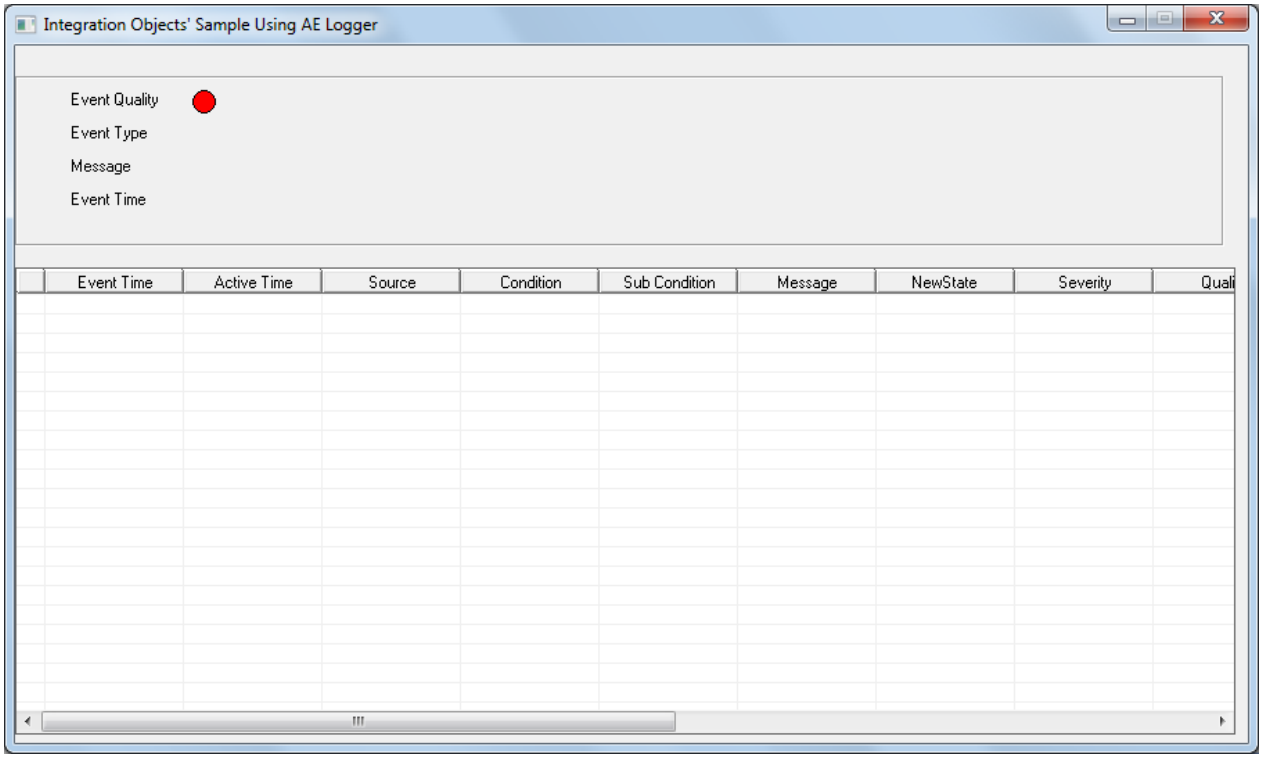
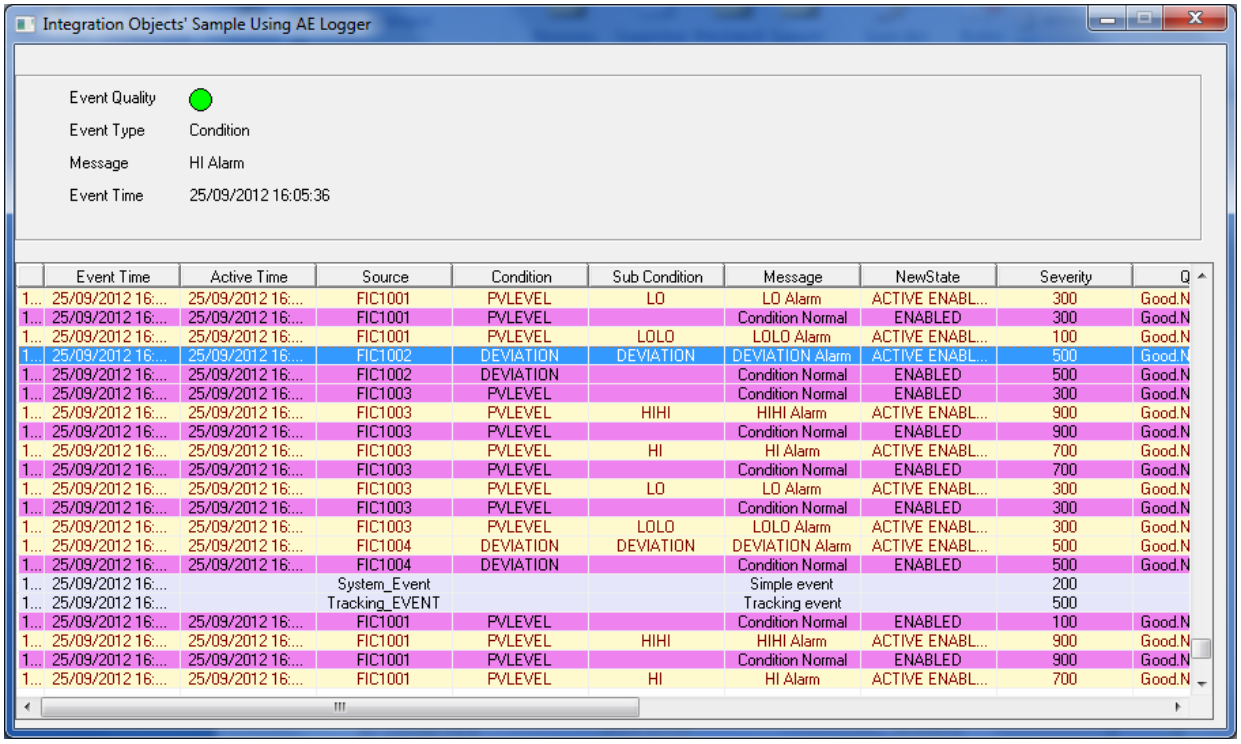


Figure 14: Integration Objects' Alarms Logger Inserted into the VB Form



	Event Time	Active Time	Source	Condition	Sub Condition	Message	NewState	Severity	Q
1...	25/09/2012 16:...	25/09/2012 16:...	FIC1001	PVLEVEL	LO	LO Alarm	ACTIVE ENABL...	300	Good.N
1...	25/09/2012 16:...	25/09/2012 16:...	FIC1001	PVLEVEL		Condition Normal	ENABLED	300	Good.N
1...	25/09/2012 16:...	25/09/2012 16:...	FIC1001	PVLEVEL	LOLO	LOLO Alarm	ACTIVE ENABL...	100	Good.N
1...	25/09/2012 16:...	25/09/2012 16:...	FIC1002	DEVIATION	DEVIATION	DEVIATION Alarm	ACTIVE ENABL...	500	Good.N
1...	25/09/2012 16:...	25/09/2012 16:...	FIC1002	DEVIATION		Condition Normal	ENABLED	500	Good.N
1...	25/09/2012 16:...	25/09/2012 16:...	FIC1003	PVLEVEL		Condition Normal	ENABLED	300	Good.N
1...	25/09/2012 16:...	25/09/2012 16:...	FIC1003	PVLEVEL	HIHI	HIHI Alarm	ACTIVE ENABL...	900	Good.N
1...	25/09/2012 16:...	25/09/2012 16:...	FIC1003	PVLEVEL		Condition Normal	ENABLED	900	Good.N
1...	25/09/2012 16:...	25/09/2012 16:...	FIC1003	PVLEVEL	HI	HI Alarm	ACTIVE ENABL...	700	Good.N
1...	25/09/2012 16:...	25/09/2012 16:...	FIC1003	PVLEVEL		Condition Normal	ENABLED	700	Good.N
1...	25/09/2012 16:...	25/09/2012 16:...	FIC1003	PVLEVEL	LO	LO Alarm	ACTIVE ENABL...	300	Good.N
1...	25/09/2012 16:...	25/09/2012 16:...	FIC1003	PVLEVEL		Condition Normal	ENABLED	300	Good.N
1...	25/09/2012 16:...	25/09/2012 16:...	FIC1003	PVLEVEL	LOLO	LOLO Alarm	ACTIVE ENABL...	300	Good.N
1...	25/09/2012 16:...	25/09/2012 16:...	FIC1004	DEVIATION	DEVIATION	DEVIATION Alarm	ACTIVE ENABL...	500	Good.N
1...	25/09/2012 16:...	25/09/2012 16:...	FIC1004	DEVIATION		Condition Normal	ENABLED	500	Good.N
1...	25/09/2012 16:...		System_Event			Simple event		200	
1...	25/09/2012 16:...		Tracking_EVENT			Tracking event		500	
1...	25/09/2012 16:...	25/09/2012 16:...	FIC1001	PVLEVEL		Condition Normal	ENABLED	100	Good.N
1...	25/09/2012 16:...	25/09/2012 16:...	FIC1001	PVLEVEL	HIHI	HIHI Alarm	ACTIVE ENABL...	900	Good.N
1...	25/09/2012 16:...	25/09/2012 16:...	FIC1001	PVLEVEL		Condition Normal	ENABLED	900	Good.N
1...	25/09/2012 16:...	25/09/2012 16:...	FIC1001	PVLEVEL	HI	HI Alarm	ACTIVE ENABL...	700	Good.N

Figure 15: Integration Objects' Alarms Logger Retrieving Real-Time Alarms

Integration Objects' OPC Event Client ActiveX allows the user to implement an AE Client using a set of methods and properties.

CONFIGURING THE AE LOGGER

1. Design-Time Configuration

Integration Objects' Alarms Logger includes property pages that enable you to easily configure the product to connect to an OPC AE Server and to retrieve incoming Alarms and Events. To view or modify the Alarm Logger properties, right click on the control and choose Properties. The dialog in Figure 14 will appear.

Below are screenshots of the control properties dialog and a description of its use.

1.1. OPC Options Configuration

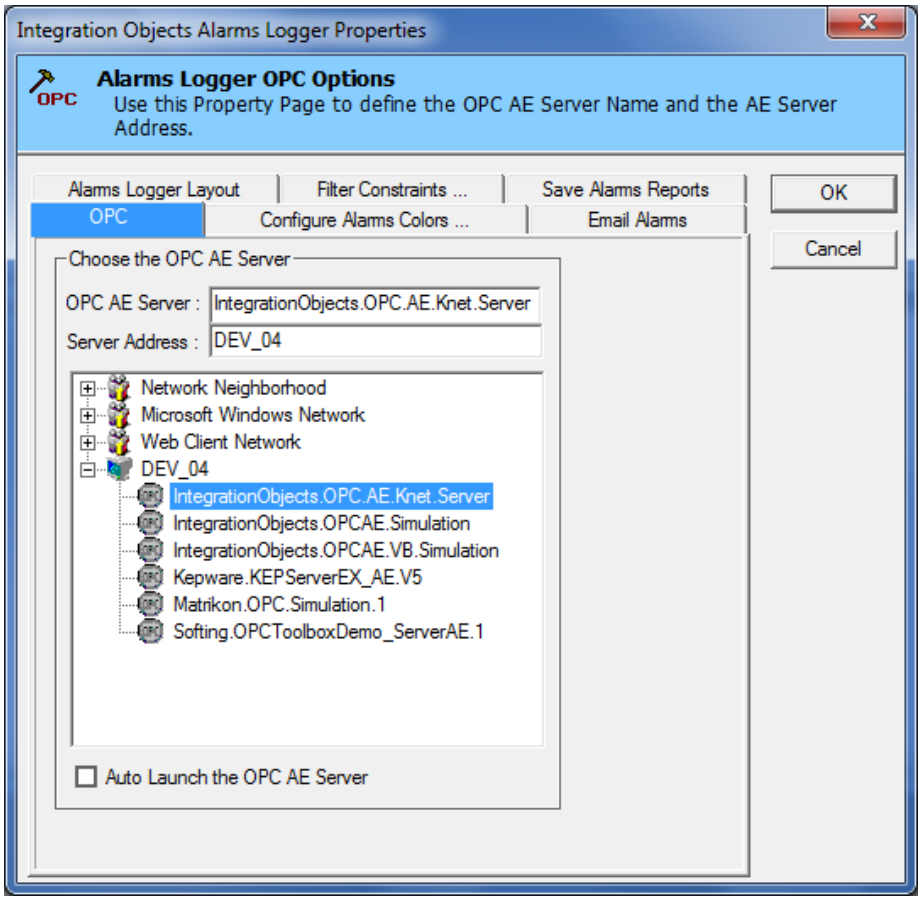


Figure 16: Alarms Logger OPC Options

Figure 14 represents the first tab sheet of the properties dialog, which is used to define the OPC AE Server and the AE Server location. The tree-view in the tab allows the user to browse the server he wants.

To pick up an AE server, you can browse the registered AE servers by simply double clicking on a machine node. You can also enter the AE server name and the server address in the edit boxes (Figure 15).

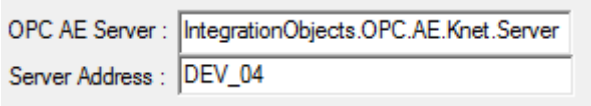


Figure 17: Entering AE Server Settings

1.2. OPC Alarm Color Configuration

Integration Objects' Alarm Logger offers several options to configure GUI to be efficient and to respect the users' preferences. To change the configuration, use the "Configure ALARMS Colors" tab sheet in the main properties window.

The next tab sheet gives users the possibility to adjust background and text colors for alarms. Colors are an important piece of information for the alarm display.

You may also choose to enable blinking of active and unacknowledged alarms by clicking on the "enable blinking" check box.

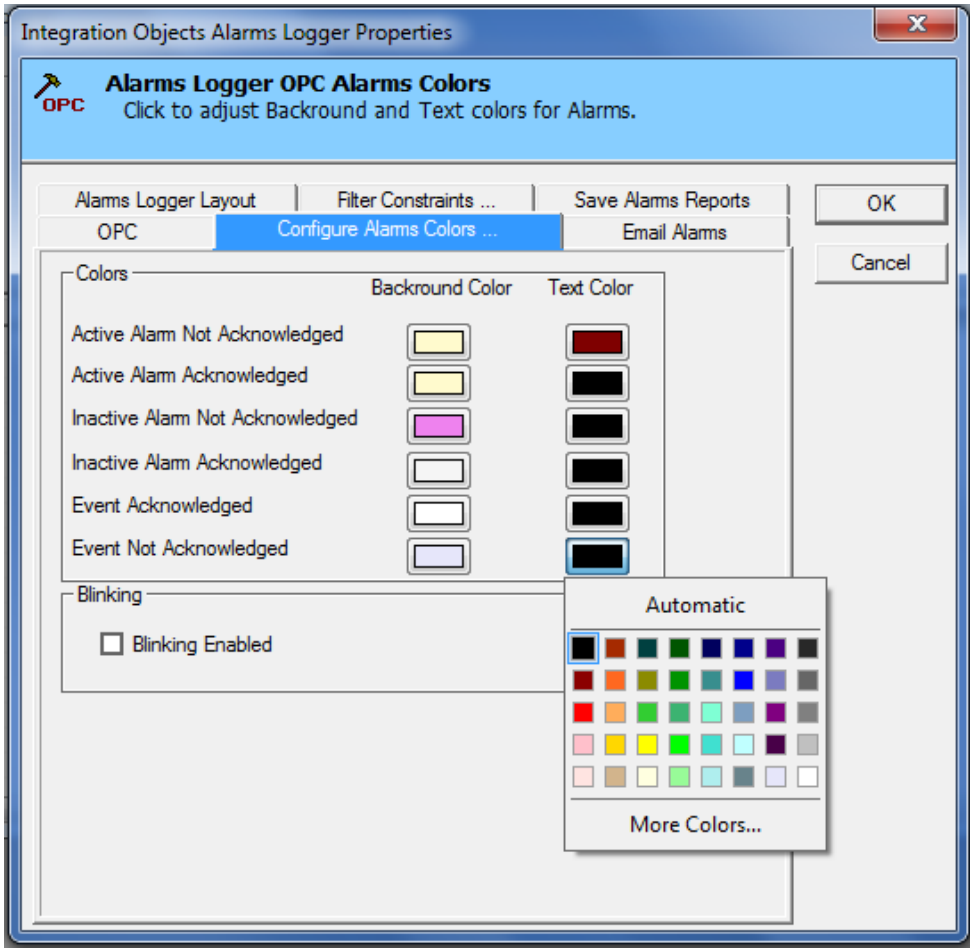


Figure : Enable Blinking

1.3. Email Alarms Configuration

You can configure the application to send emails when an event or an alarm occurs. The user must provide the destination email address and SMTP server.

Integration Objects' Alarm Logger has a tab sheet used to configure email notifications. When an alarm occurs, the application using the control automatically sends detailed notifications to the administrator by e-mail. The user can specify low or high severity for received alarms and events.

To enable mail notification, you should check the "enable mail notification" check box. If you choose to select the "send only critical alarms" option, you will receive only the incoming alarms that meet the severity level you define.

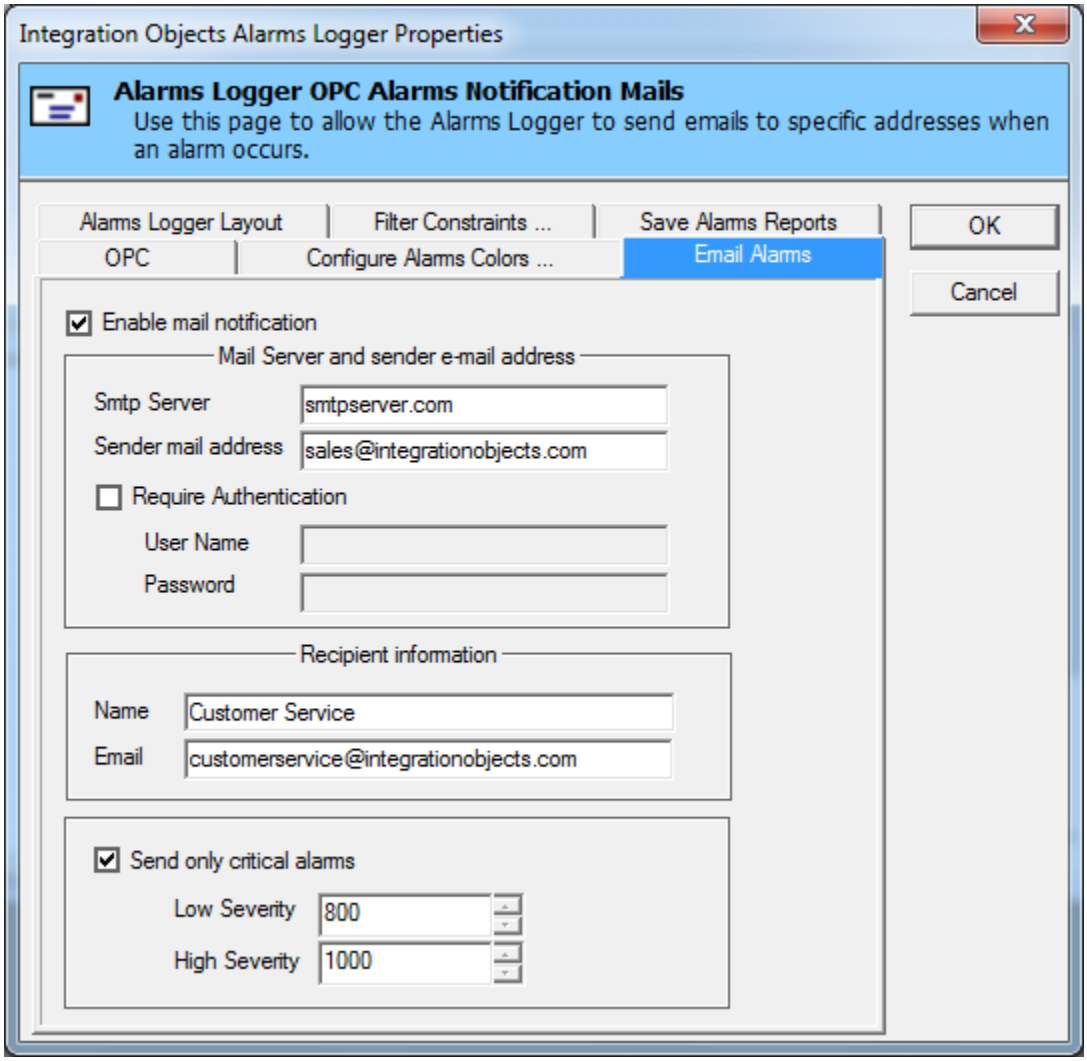


Figure 18: Configure Email Notifications

1.4. Control Layout Configuration

The Control layout tab is used to select the columns you want to appear on your control panel. The following dialog box allows users to view custom information, to pick up columns you want to display in the control panel, and to choose displayed format data to access further information.

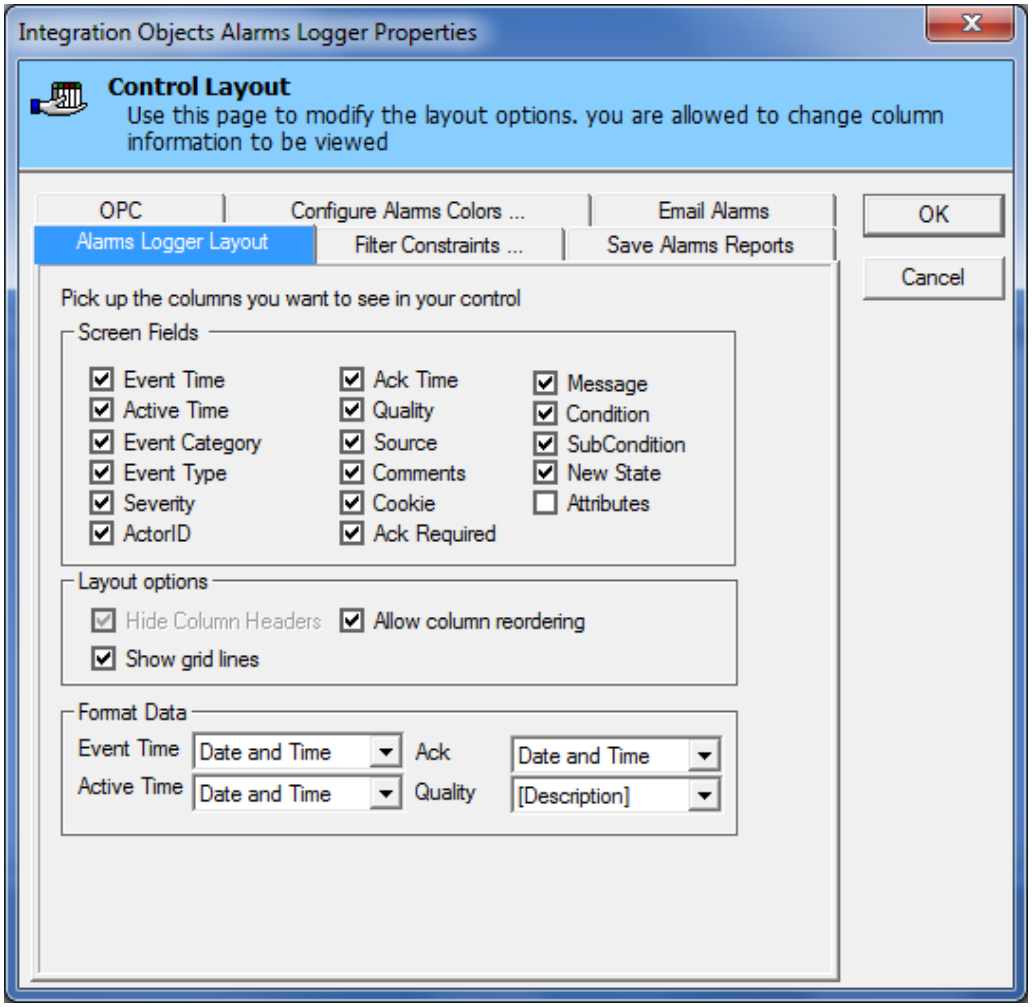


Figure 19: Alarms Logger Layout

This ActiveX control includes the following columns:

Column Heading	Description
Event Time	The time that the event was generated.
Active Time	The time that the event was active.
Source	The source item that caused the event to occur.
Condition	The condition name that caused the event to occur.
Sub Condition	The detailed cause of the event.
Message	The alarm message.
Event type	OPC specific event type (simple, tracking, condition).
Event category	One of the OPC event categories.
Quality	An indicator to the reliability of the event.

Actor ID	The name of the user that acknowledged the event.
Ack Comment	The comments entered by the user when he acknowledges the event.
Ack required	Indicates if the event requires acknowledgment.
Severity	The severity of an alarm.
Ack Time	The time when the user acknowledged the alarm.
Cookie	
NewState	The type of alarm (active, acknowledged …).
Attributes	Vendor specific server attributes.

Table 2: The ActiveX Columns

If the user wants to pick up only some screen fields to display in the control, simply uncheck the unwanted columns.
All columns are checked by default except the “Attributes” column which the user may mark if so desired.

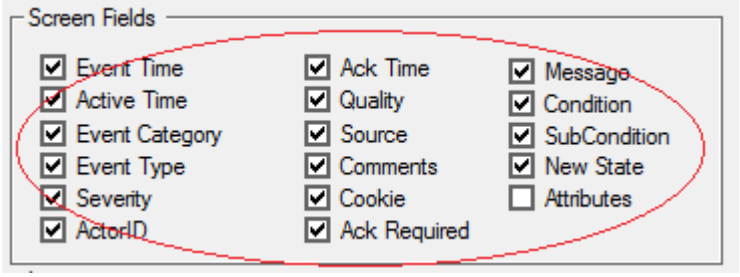


Figure 20: Pick Up Columns to Display in the Logger

This Layout dialog allows users to define layout options for the control.
The user can choose the ability to drag-and-drop column headers to reorder columns in the control. By default, drag-and-drop reordering of columns is enabled (Figure 20).

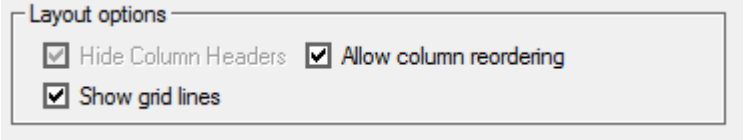


Figure 21: Enabling Reordering Columns

To move a column, click the column header once to select it. Then click and drag the column header to a new location (Figure 21).

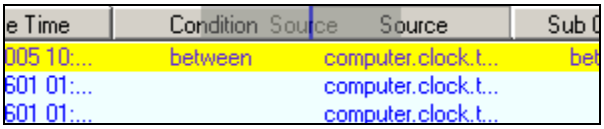


Figure 22: Drag-and-Drops Column Headers to Reorder Columns

We've decided to add a new preference on the layout page which allows you to select the time format for some screen field controls.

Along with choosing a custom date/time format, the user may choose to display date and time or just time for the three fields: **Event Time**, **Active Time** and **Ack Time**.

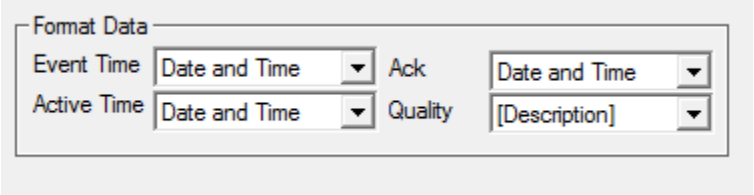


Figure 23: Customizing Data

1.5. Filter Constraints Configuration

The Filter constraints tab is used to set criteria for including or excluding specific alarms in the list view.

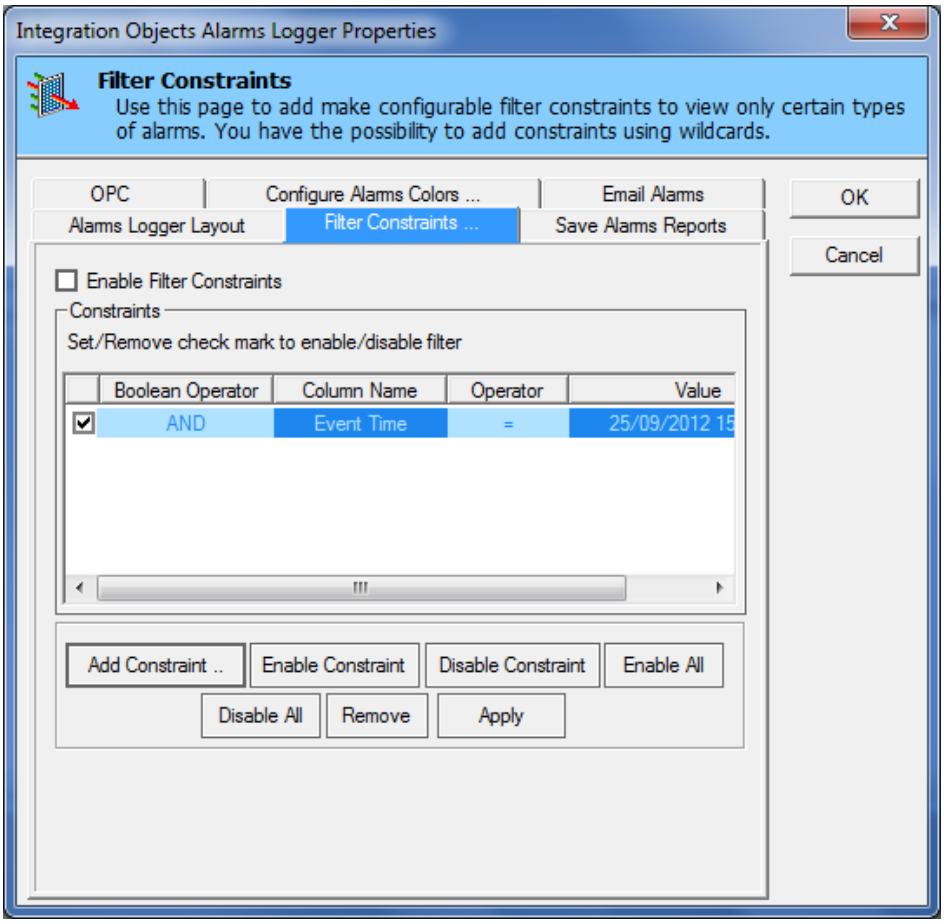


Figure 24: Filter Constraints

To define a filter constraint, click the *Add Constraint* button and the dialog (Figure 24) will appear. You must select the filter column name, the operator (=, #, <, …) and the value.

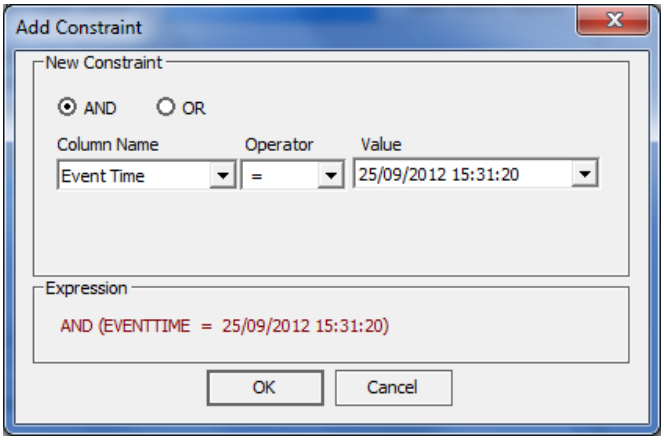


Figure 25: Add Constraint Dialog

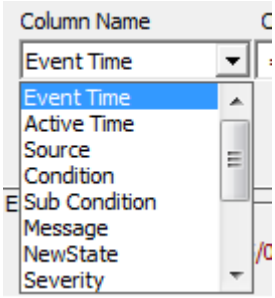


Figure 26: Select the Constraint Field

Figure 27: Select the Constraint Operator

In some kind of filter constraints, the user has the possibility to enter string values using wildcards (?, *, …).

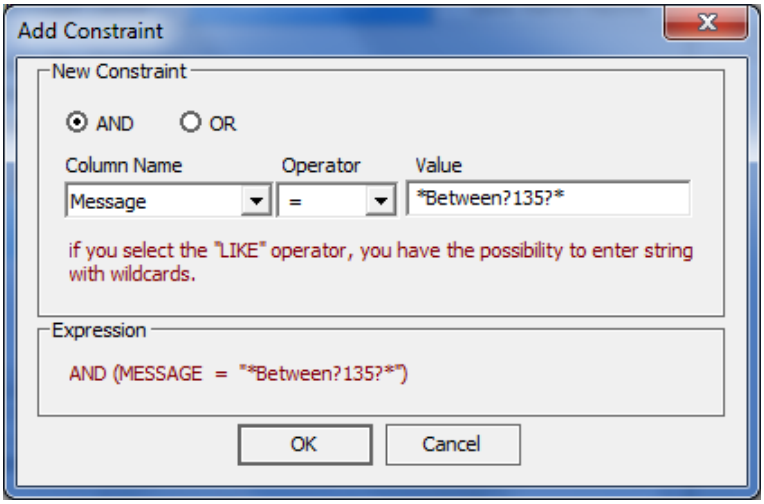


Figure 28: Constraint Using Wildcards

Before moving to another tab, you should click the *Apply* button to save modifications. If you did not do so, the following message box will appear.

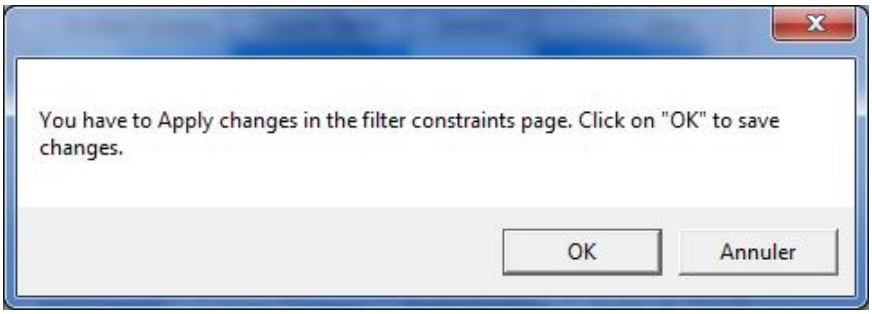


Figure 29: Apply Changes Notification Message Box

1.6. Save Alarm Reports Configuration

Integration Objects' Alarms Logger allows the user to save the retrieved alarms into report files. To do that, the user has the possibility to define how frequently he wants the save to be performed (each hour or after a certain number of hours).

By default, alarms are saved each hour into the directory where the application deploying the control is located. If you are not pleased with this location, you can select another directory. Click *Browse Path* to select the folder where you want to save the alarms. When saving the alarms report, the toolkit creates a file in the chosen folder which is entitled: "AlarmsReports{DAY]_{MONTH]_[YEAR]_{HOUR]_{MIN]_{SECONDS]" with the "csv" extension.

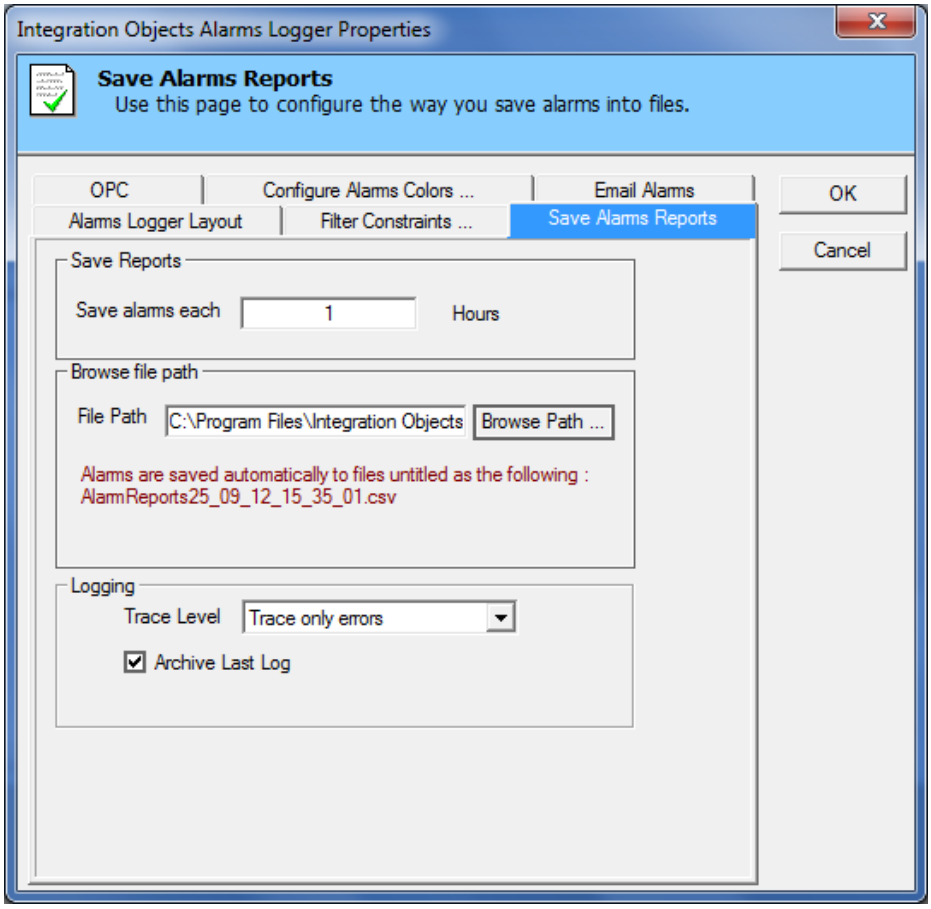


Figure 30: Save Alarms Reports

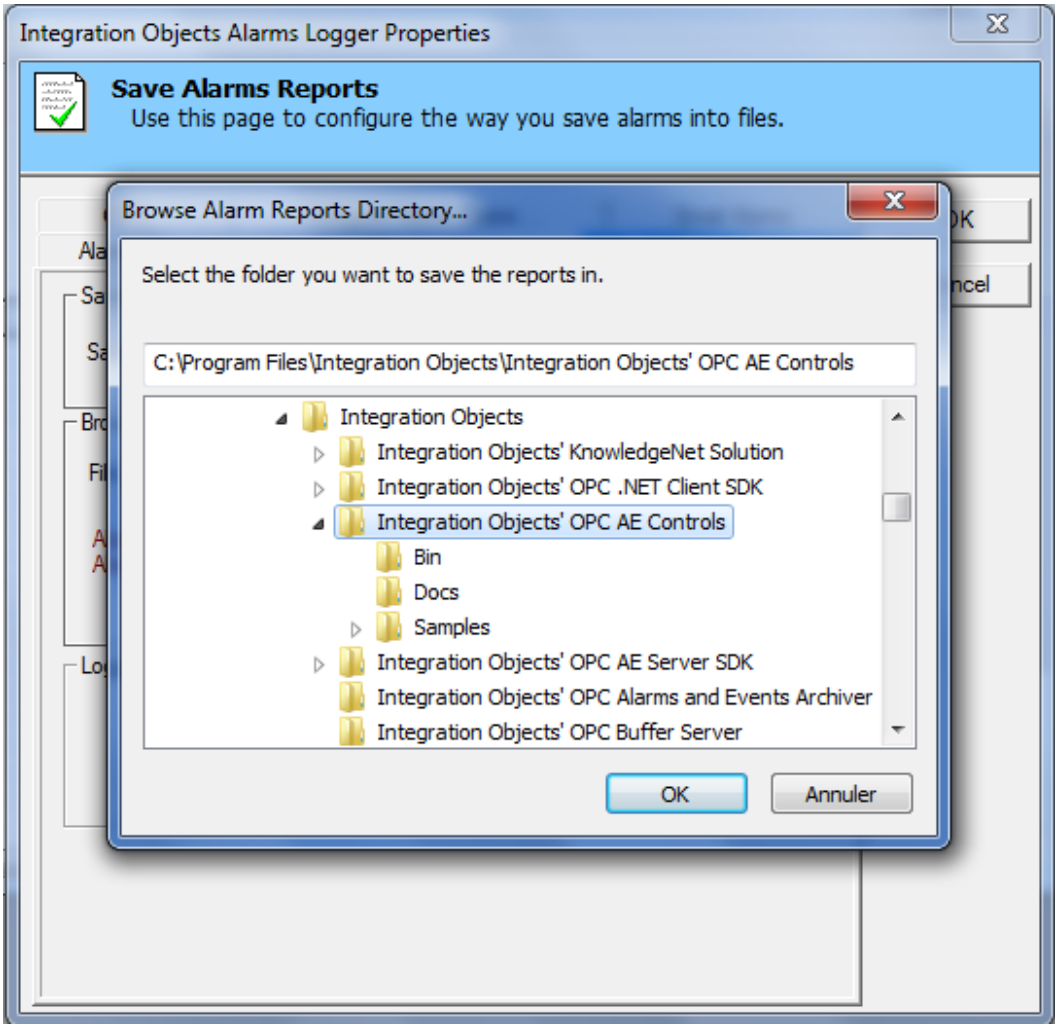


Figure 31: Select Reports Folder

1.7. Logging

Integration Objects' Alarms Logger produces a log file named "IOAEOCX_LogEvent.LOG" that records errors and debugging information in design time. It also produces a second file named "IOAEOCX_LogEvent1.LOG" that records errors and debugging information at runtime. If difficulties occur with the application, the log file can be extremely valuable for troubleshooting. When operations are running normally, the control will log very little information.

These log files are generated by default in the bin folder under the installation folder. This path can be modified by the user using the "Configuration.ini" configuration file incorporated by the toolkit which includes several logging parameters and a parameter used to define the folder where the application saves retrieved alarms. These parameters all have default settings and can be changed at start-up by editing the configuration file, or by simply changing the information in the control properties window.

To change this file:

1. Open Configuration.ini in a text editor.
2. Edit any of the parameters listed in the following tables:

Log Setting	Description	Default Value
LogFileMaxSize	The maximum log file size, in bytes. Once this size is reached during run-time, the log file is overwritten.	1048576*2 ~ 2 Mb (MegaByte)
TraceLevel	The trace level is a value telling what type of information to log: 0: Only errors messages are logged. 1: Some extra information. 2: Debugging information is logged. 3: Detailed debugging information. The higher the trace level, the more information is recorded. We recommend you to use level 0 for a better performance of the client application.	0
ArchiveLastLog	TRUE: Old file is copied to an intermediate file with incremental extension, before being overwritten. FALSE: Any pre-existing log file is erased and overwritten at start-up.	FALSE
ReportsFolder	The path where the alarms report will be saved	C:
LogFilePath	The path where the log file will be generated. It's set to the bin folder by default and can be modified by the user to change it to a customized one.	The Bin folder path

Table 3: Configuration File Properties

However, it's much easier and safer to change the "ReportsFolder" settings in the Save Alarms Reports tab, instead of editing the .ini file yourself. We strongly suggest that you do not change the values in these optional sections of the .ini file, but instead, make any needed changes using the interface.

If you would like to automatically archive log files, select the **Archive last log** check box.

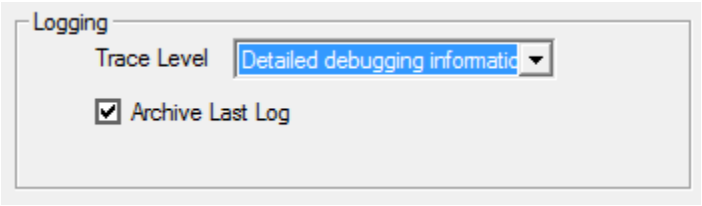


Figure 32: Log Configuration Dialog

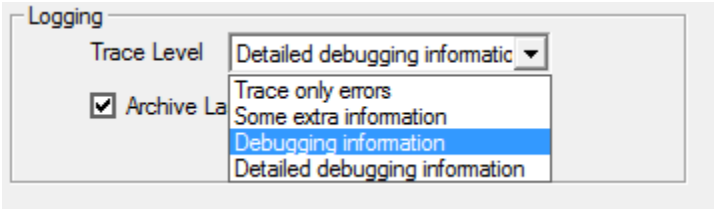


Figure 33: Choosing the Log Level

2. Runtime Configuration

When running the application deploying the Integration Objects' Alarms Logger, if you right click on the control, the following menu appears (figure 30):

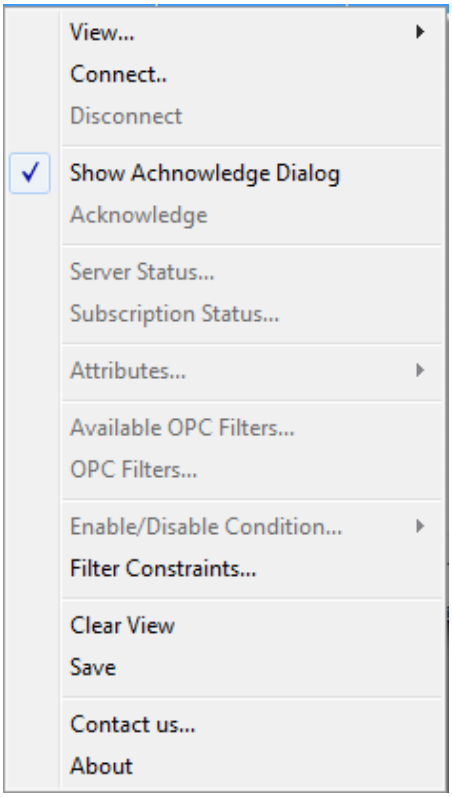


Figure 34: Context Menu

The first popup menu appears when the application client is not connected to an OPC AE Server.

If you want to pick up the screen fields you want to view in the control layout, just click the *View...* menu item (Figure 31).

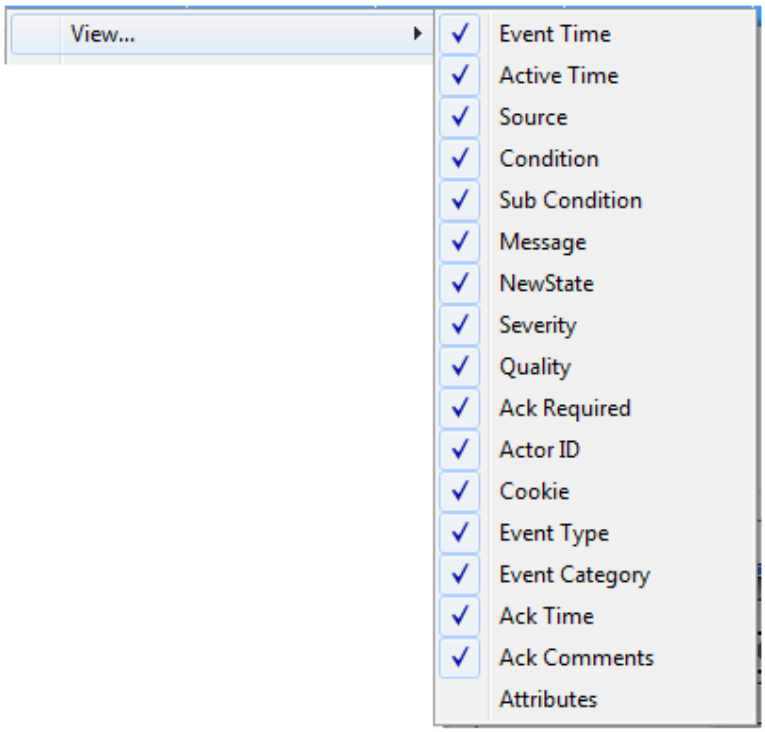


Figure 35 View Menu Item

2.1. Connect to an OPC AE Server

To connect to an AE Server at runtime, choose the menu item “Connect…” from the popup menu. The following window will be displayed.
Browse your network to select the computer node on which the server is located and the ProgID AE server, and then click *Connect*.

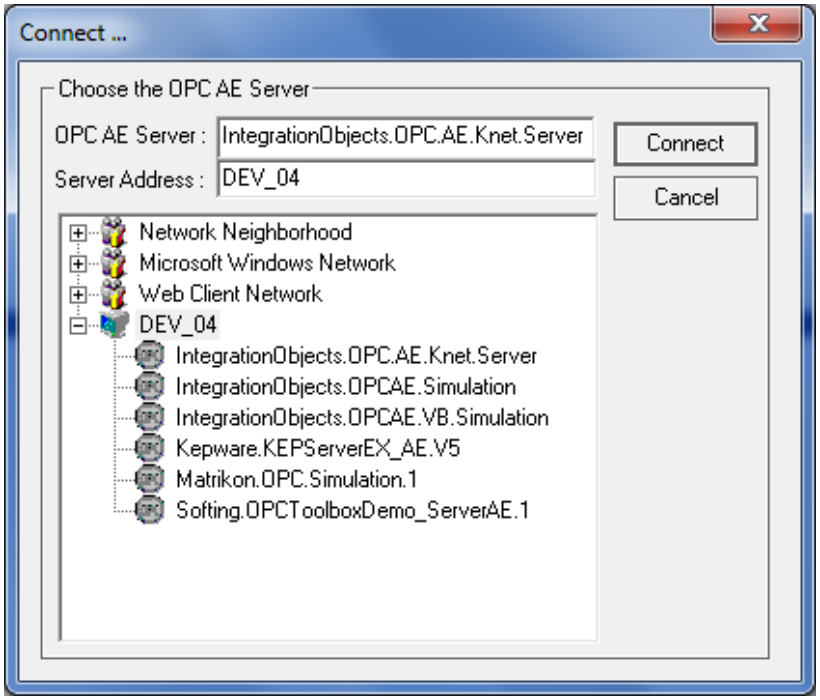


Figure 36: "Connect" from Context Menu

2.2. Disconnect from an AE OPC Server

If your application is connected to an OPC AE Server and you want to disconnect from it, click *Disconnect* from the popup menu.

2.3. Acknowledge Alarms

This section provides information on acknowledging alarms:

To acknowledge an alarm requiring a response, the user may double click on the entry, or click the *Acknowledge* menu item from the context menu.

When the user right clicks on an alarm and chooses Acknowledge, the following dialog box will appear.

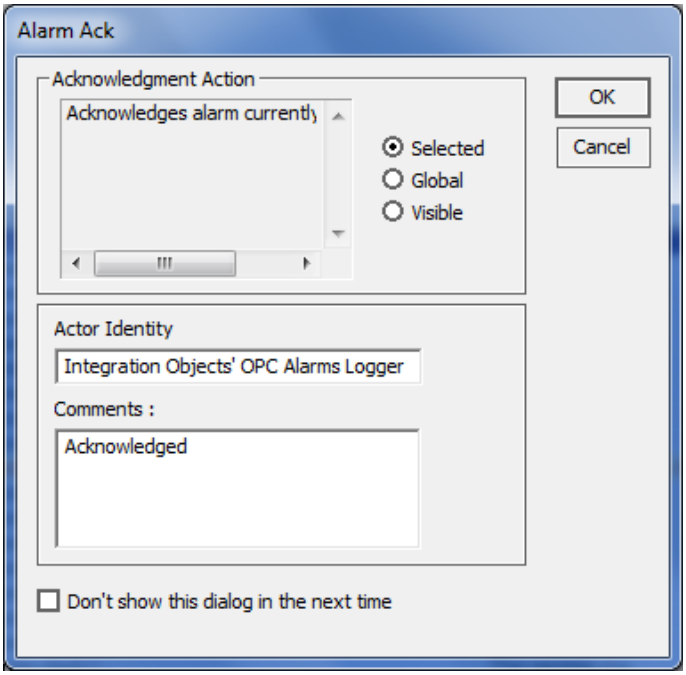


Figure 37: Acknowledging Alarms

In the Alarm Acknowledgment dialog box, users have the possibility to choose between three acknowledgment options which are listed in the table below.

Acknowledgment Options	Description
Point	Acknowledges alarms currently selected.
Global	Acknowledges all alarms received. This option quickly acknowledges all alarms from the current view.
Visible	Acknowledges all visible alarms. For example, if the size of the Viewer shows five alarms, and a total of eight alarms came in, only the five visible alarms are acknowledged.

Table 4: Acknowledgment Options

Users can also add comments and change the actor identity. If the user wants to always have the same acknowledgment options, he can click the *Don't show this dialog the next time* check box.

To display the acknowledgment options dialog once again, he may mark the "Show Acknowledge dialog" from the popup menu.

2.4. Server Information

To view the OPC server status, choose the “Server Status…” menu item in the popup menu (Figure 30) and the server status dialog box will appear (Figure 34). This window shows all static information about the AE Server.

The server status includes:

- Server start time
- Current time
- Time of last update sent
- Current OPC server state
- Major version of server
- Minor version of server
- Build number of the server
- Vendor specific information

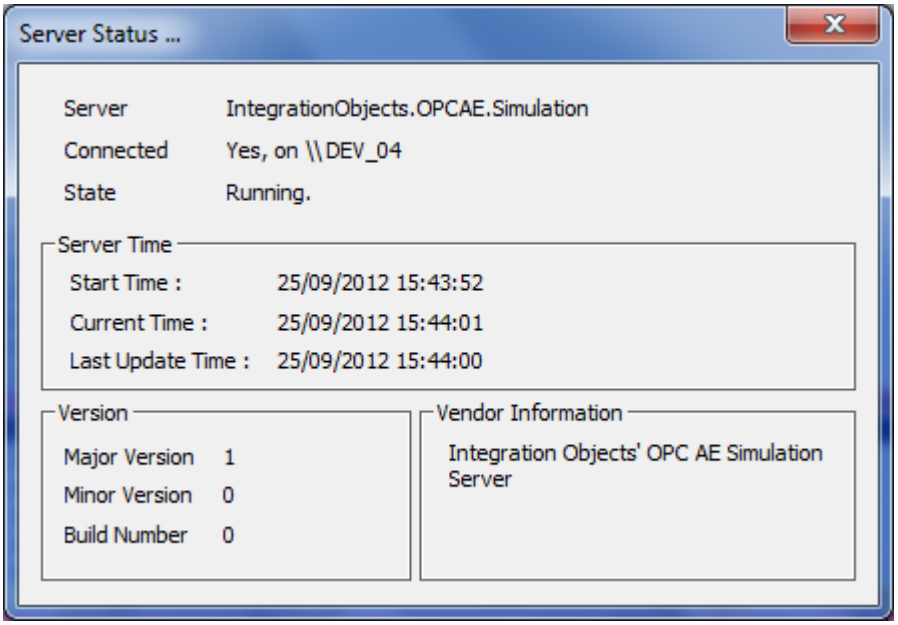


Figure 38: Server Status

2.5. Event Subscription Status

To view the Event Subscription status, choose the “Subscription Status…” menu item in the context menu and the following dialog is displayed. This dialog allows users to get subscription state information and to modify the subscription state.

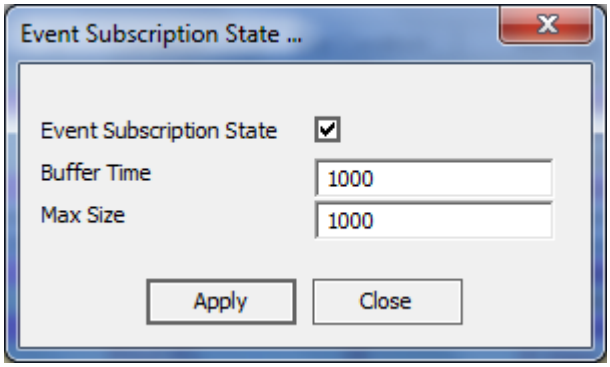


Figure 39: Event Subscription Status

The event subscription state is presented as a check box. When checked, the event subscription is active and it sends OnEvent notifications.

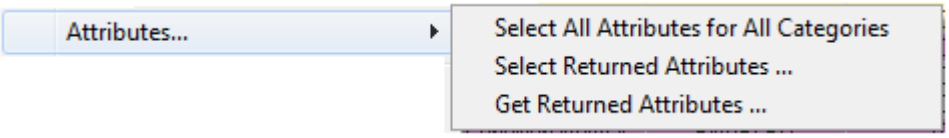
The buffer time is the time in milliseconds that the server can hold back the notification to buffer multiple events. Example, 1000 would indicate that the server buffer events along 1 second and sends the whole set of events as one callback.

The max size is the maximum number of events to buffer before sending events.

2.6. Select Returned Attributes

To retrieve the attributes of an existing Event Subscription, the user should select the Select Returned Attributes menu item. For each event category, SelectReturnedAttributes picks out the attributes to return. This method can be called many times in order to specify the attributes to return for each unique event type and event category pair. If this is called multiple times for the same event type and event category pair, then the latest call will be considered.

If you choose the “Select All Attributes for All Categories”, all server event attributes will be displayed.



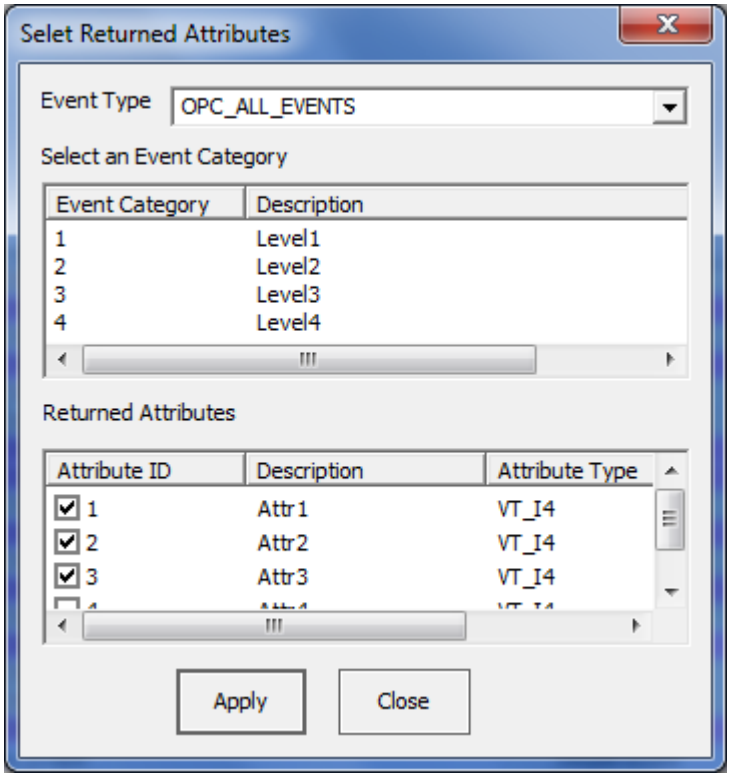


Figure 40: Select Specific Server Returned Attributes

2.7. Retrieve Returned Attributes

To get the attributes, the user should click the *Get Returned Attributes* menu item. For each event category, the following dialog retrieves the attributes previously specified by the user in the Select Returned Attributes dialog.

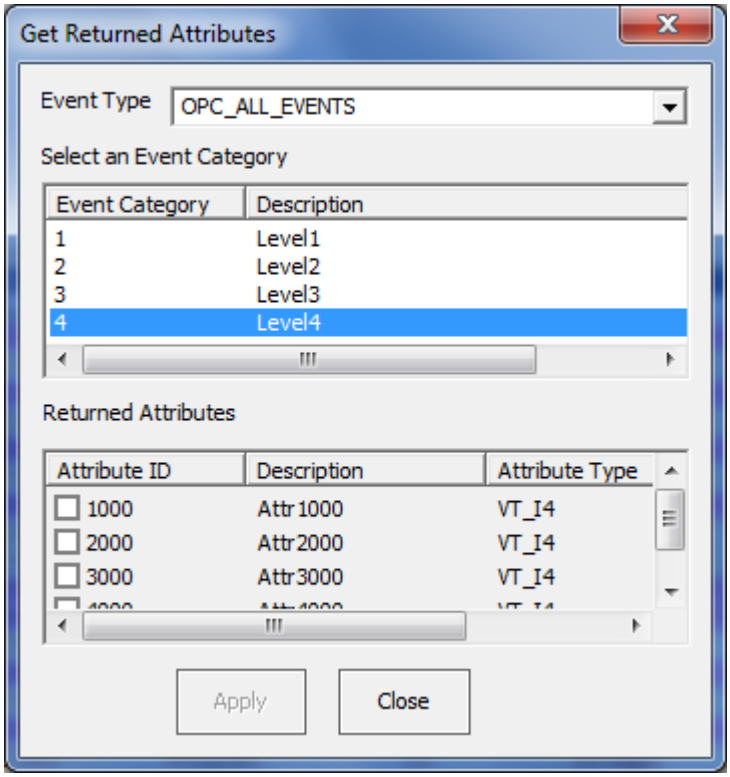


Figure 41: Get Specific Server Returned Attributes

2.8. Enable/Disable condition by areas and sources

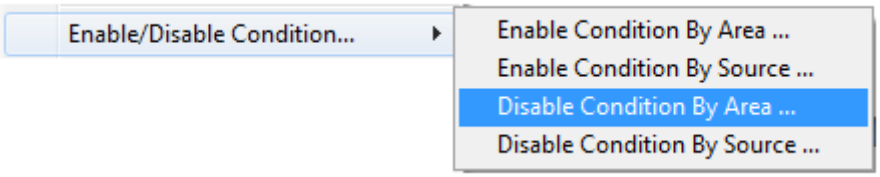


Figure 42: Enable/Disable Condition Menu Item

Enable condition by area: Enable condition alarm events in an area.
Enable condition by source: Enable condition alarm events for a given source.
Disable condition by area: Disable condition alarm events in an area.
Disable condition by source: Disable condition alarm events for a given source.

The dialog box below is the “enable condition by source.” The user should navigate into the tree and check the wanted sources names in order to enable conditions for a given source.

The similar thing should be done if the user wants to enable a condition by area or to disable a condition by source or by area.

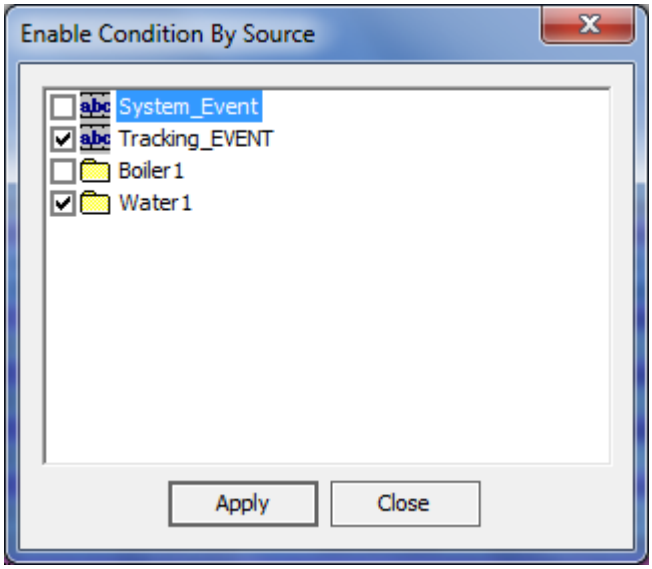


Figure 43: Enable Condition by Source

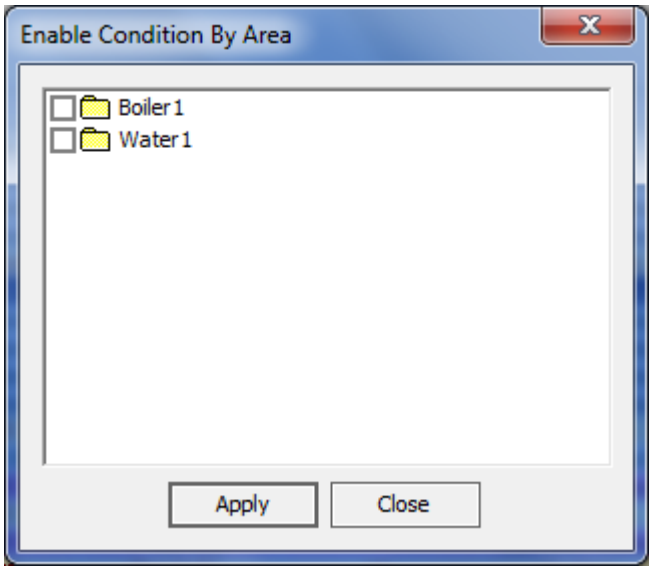


Figure 44: Enable Condition by Area

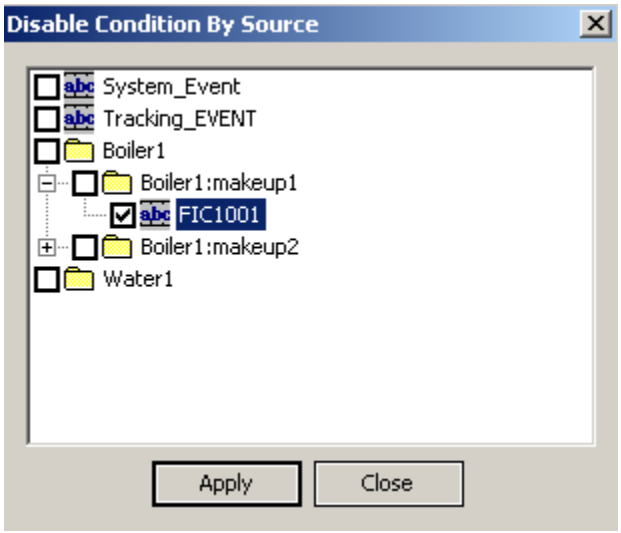


Figure 45: Enable Condition by Area

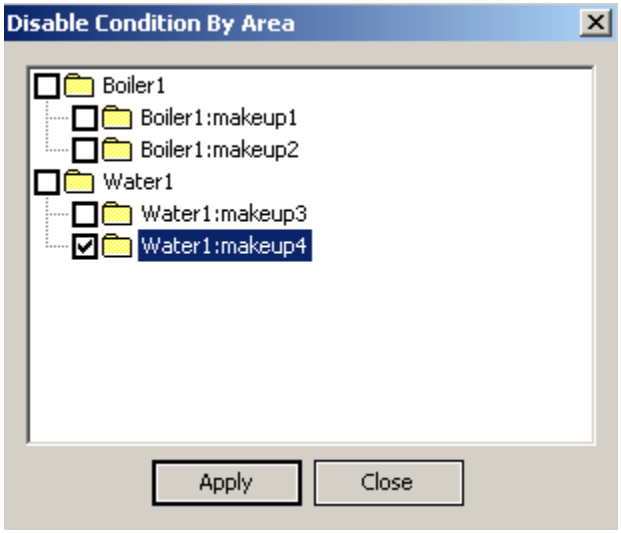


Figure 46: Enable Condition by Area

2.9. Available Filters

To view the filters of an OPC alarms and events server, the user should select the *Available OPC Filters* menu item and the following dialog screen appears:

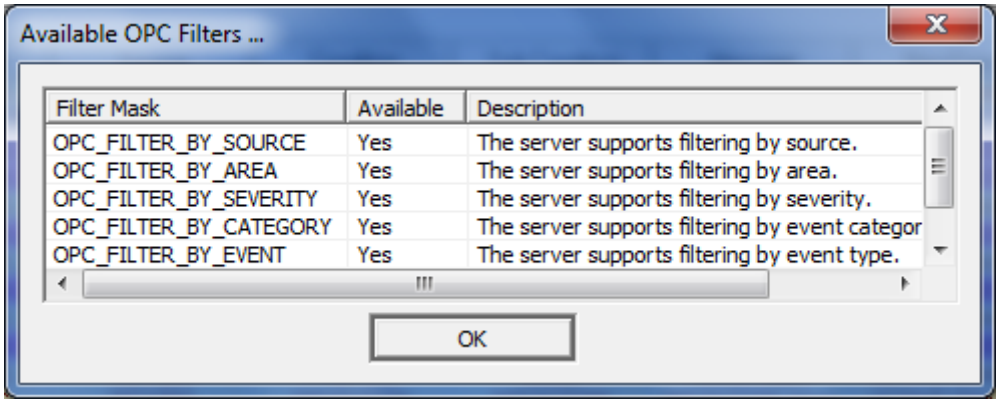


Figure 47: Available Filters

2.10. OPC Filtering Configuration

To define OPC Filters, click the *OPC Filters* menu item from the context menu. The OPC Filtering dialog is used to set criteria for including or excluding specific alarms in the control. Users can specify the event types, severity, event category, areas or sources to be filtered.

The user can set filters on any event subscription in order to limit the events that he will be notified of. To setup a filter for an event subscription, the user can use the following criteria:

- **Filtering by event type**: only events satisfying the criterion “Event Type” will be returned.
- **Filtering by event categories**: only events satisfying the criterion “Event Categories” will be returned.
- **Filtering by areas and sources**: only events satisfying the criterion “Existing in these areas or having these sources” will be returned.
- **Filtering by Severity**: only events satisfying the criterion “Events that have a severity between the min and the max severity” will be returned.

Users can select multiple criteria; they will be logically ANDed together. This will cause all events satisfying these selected criteria to be returned.

If the user wants to get only specific event alarms, he may select the wanted check boxes (Figure 44).

The list of the event categories on the left indicates the categories to be filtered. The events on the right are the available categories. Select an event category and click on buttons (>> and <<) to move categories from one area to the other.

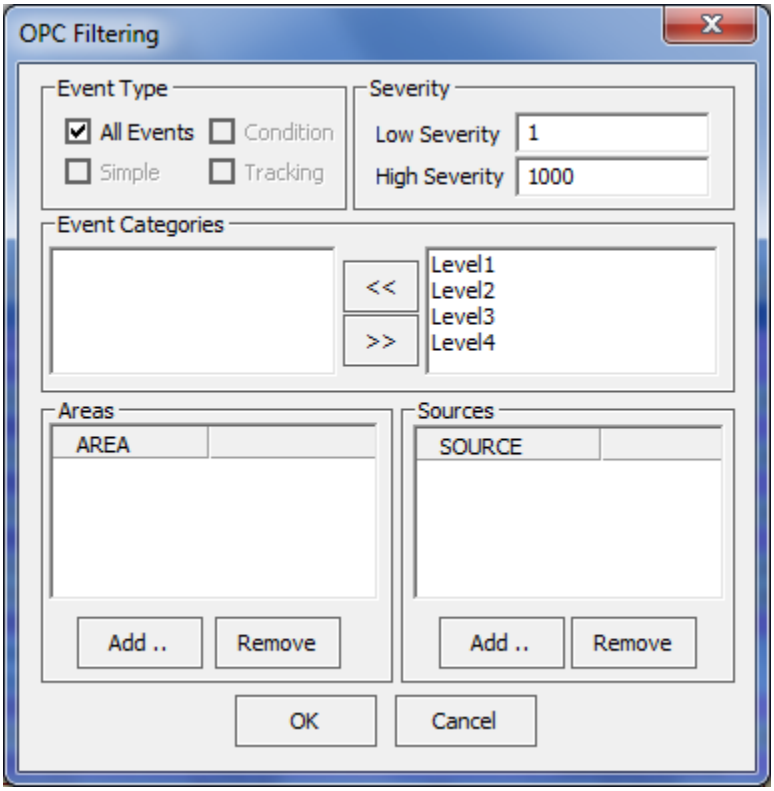


Figure 48: Configure OPC Filtering Alarms

The list of areas indicates the areas containing the source names needing to be filtered. To add a new area or a new source, click the *Add* button and a dialog box appears (Figure 43 and Figure 46). The user may remove an area or a source by clicking the *Remove* button.

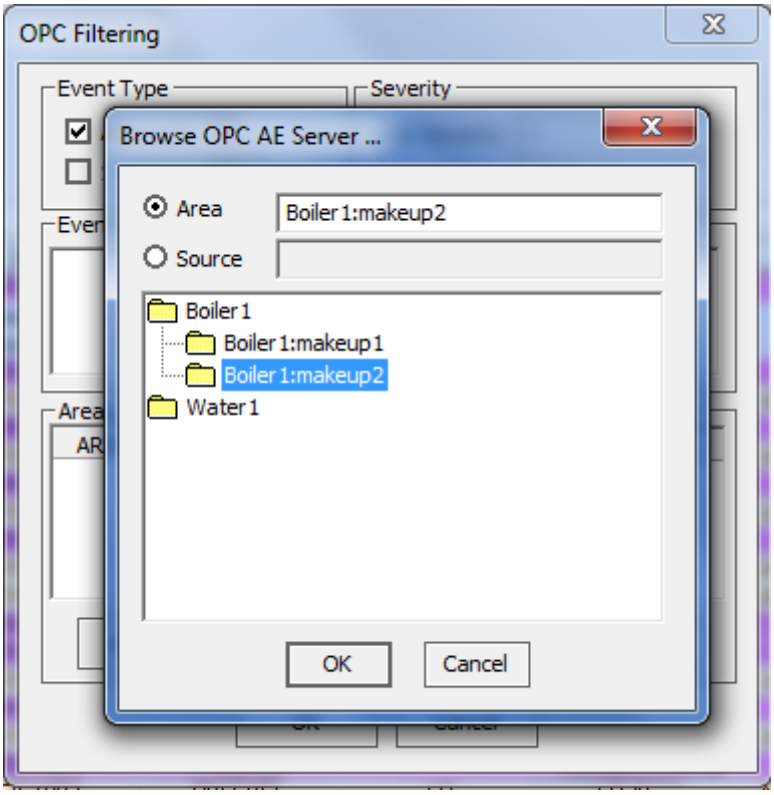


Figure 49: Add Areas

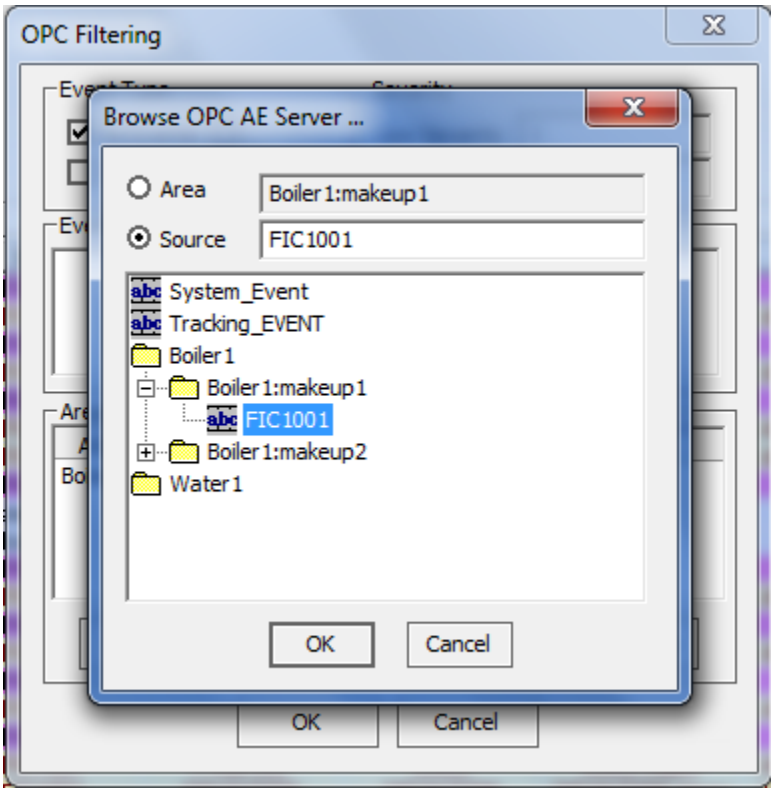


Figure 50: Add Sources

2.11. Apply Filter Constraints

By default, the AE Logger is configured to display all incoming alarms. To customize which alarms are displayed, right-click on the control, select *Filter Constraints…* and build an equation to restrict only the alarms that respond to the build filter constraint.

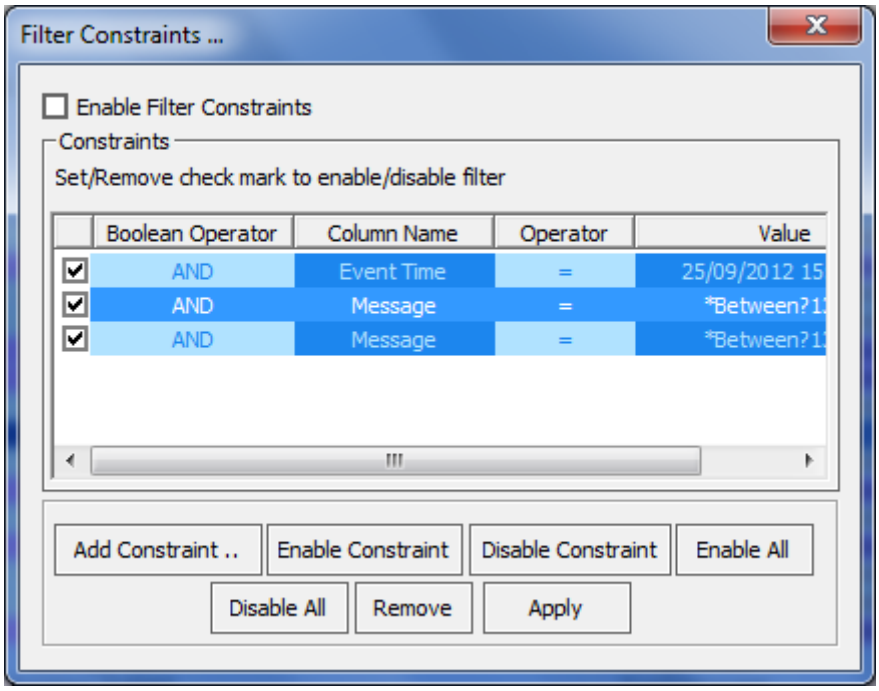


Figure 51: Filter Constraints Window at Runtime

2.12. About Box Dialog

Click *About* from the menu to display the About Box for the control. The About Box contains the product name and version number as well as other information about the software and Integration Objects.

Figure 52: About Box

3. Control Properties and Methods

To edit the control properties, the user can simply use the control properties wizard and the runtime context menu without having to write any line code.

Property ServerName:
Description: The OPC AE Server Name you want to connect to.
Syntax: Property Public ServerName As String
Example:

AlarmsLogger .ServerName = "IntegrationObjects.OPCAE.Simulator"

Property ServerAddress
Description: The OPC AE Server location.
Syntax: Property Public ServerAddress As String
Example:

AlarmsLogger .ServerAddress = "\\MyServer"

Method Connect
Description: Connect to the defined Server.
Syntax : Public Function Connect() As Long
Example:

ServerName = "IntegrationObjects.OPCAE.Simulator"
ServerAddress = "\\127.0.0.1"
AlarmsLogger.Connect()

Method Disconnect

Description: Disconnect from the OPC Server.
Syntax : Public Function Disconnect() As Long
Example:

```
AlarmsLogger.Disconnect()
```

4. Control Events

4.1. OnEvent event

This event is posted to the application deploying the ActiveX whenever the control receives a message from the connected AE server. The only Argument within the event defines the event object. The event has the following syntax.

4.2. Public Event OnEvent(OpcEvent as Object)

To allow the application to obtain data from an event message posted by the connected AE Server, the user can simply double click the ActiveX control (VB environment) to view the event announcement. You may then add instructions to read the values for specific data contained within the event object.
Below is the detailed description of each data value for the event object.

4.2.1. Public Property AckRequired As Boolean

Description: True if the event requires acknowledgment.

4.2.2. Public Property ActiveTime As Date

Description: The time that the event was active.

4.2.3. Public Property EventTime As Date

Description: The time that the event was generated.

4.2.4. Public Property ConditionName As String

Description: The condition name that caused the event to occur.

4.2.5. Public Property SubConditionName As String

Description: The detailed cause of the event.

4.2.6. Public Property EventCategory As Long

Description: One of the OPC event categories.

4.2.7. Public Property Cookie As Long

Description: The unique cookie associated with the event.

4.2.8. Public Property Message As String

Description: The alarm message.

4.2.9. Public Property NewState As Long

Description: The type of alarm (active, acknowledged, …). This property is based on these values:

IO_OPCAE_ConditionStateConstants
{
IO_OPCAE_CONDITION_ENABLED
IO_OPCAE_CONDITION_ACTIVE
IO_OPCAE_CONDITION_ACKED
}

4.2.10. Public Property Quality As Long

Description: An indicator for the reliability of the event. This property is based on the following enumeration:

IO_OPCAE_QualityConstants
{
IO_OPCAE_QUALITY_BAD
IO_OPCAE_QUALITY_UNCERTAIN
IO_OPCAE_QUALITY_GOOD
}

Example:

```
Private Sub IOAlarmsLogger1_OnEvent(OpcEvent As Object)
…
If (OpcEvent.Quality =  IO_OPCAE_QUALITY_BAD)  Then
MsgBox “Bad Quality”
End If
End Sub
```

4.2.11. Public Property StrQuality As String

Description: An indicator for the reliability of the event in a string format. This value is based on these string values: (“BAD”, “UNCERTAIN”, "UNKNOWN OPC QUALITY", "GOOD","NON_SPECIFIC”)

Example:

```
        Private Sub IOAlarmsLogger1_OnEvent(OpcEvent As Object)
                        …
If (OpcEvent.StrQuality =  “BAD”)  Then
                                MsgBox “Bad Quality”
End If
End Sub
```

4.2.12. Public Property Severity As Long

Description: The severity of an alarm.

4.2.13. Public Property Source As String

Description: The source item that caused the event to occur.

4.2.14. Public Property EventType As Long

Description: OPC specific event type (simple, tracking, condition). This field is based on this enumeration:

```
IO_OPCAE_EventTypesConstants
{
IO_OPCAE_SIMPLE_EVENT
IO_OPCAE_TRACKING_EVENT
IO_OPCAE_CONDITION_EVENT
IO_OPCAE_ALL_EVENTS
}
```

Example:

```
        Private Sub IOAlarmsLogger1_OnEvent(OpcEvent As Object)
                        …
If (OpcEvent. EventType = IO_OPCAE_CONDITION_EVENT)  Then
                        …
End If
End Sub
```

4.2.15. Public Property ActorID As String

Description: The name of the user that acknowledged the event.

4.2.16. Public Property ChangeMask As Long

Description: The bits changed in the current event. The changed bits are below:

IO_OPCAE_ChangeConstants
{
IO_OPCAE_CHANGE_ACTIVE_STATE
IO_OPCAE_CHANGE_ACK_STATE
IO_OPCAE_CHANGE_ENABLE_STATE
IO_OPCAE_CHANGE_QUALITY
IO_OPCAE_CHANGE_SEVERITY
IO_OPCAE_CHANGE_SUBCONDITION
IO_OPCAE_CHANGE_MESSAGE
IO_OPCAE_CHANGE_ATTRIBUTE
}

4.2.17. Public Property EventAttributesCount As Long

Description: The number of attributes.

4.2.18. Public Property EventAttributes(Index As Integer) As Variant

Description: Vendor specific server attributes.

Example:

```
        Private Sub IOAlarmsLogger1_OnEvent(OpcEvent As Object)
If (OpcEvent. EventAttributesCount <> 0)  Then
                                Dim I as Integer
                                For I=0 to EventAttributesCount-1
                                        SelectedAttributesList.AddItem  CStr(EventAttributes(I))
                                Next
End If
End Sub
```

CONFIGURING THE .NET AE LOGGER

1. Runtime Configuration

When running the application deploying the Integration Objects' OPC AE Alarm Logger .Net ActiveX, if you right click on the control, the following menu appears (figure 53):

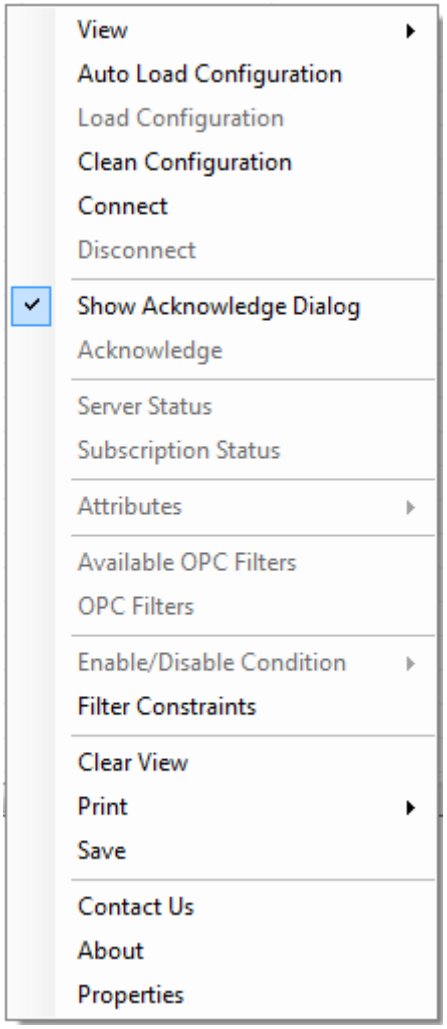


Figure 53: Context Menu

The first popup menu appears when the application client is not connected to an OPC AE Server.

4.3. View

If you want to pick up the screen fields you want to view in the control layout, just click the **View** menu item as shown below:

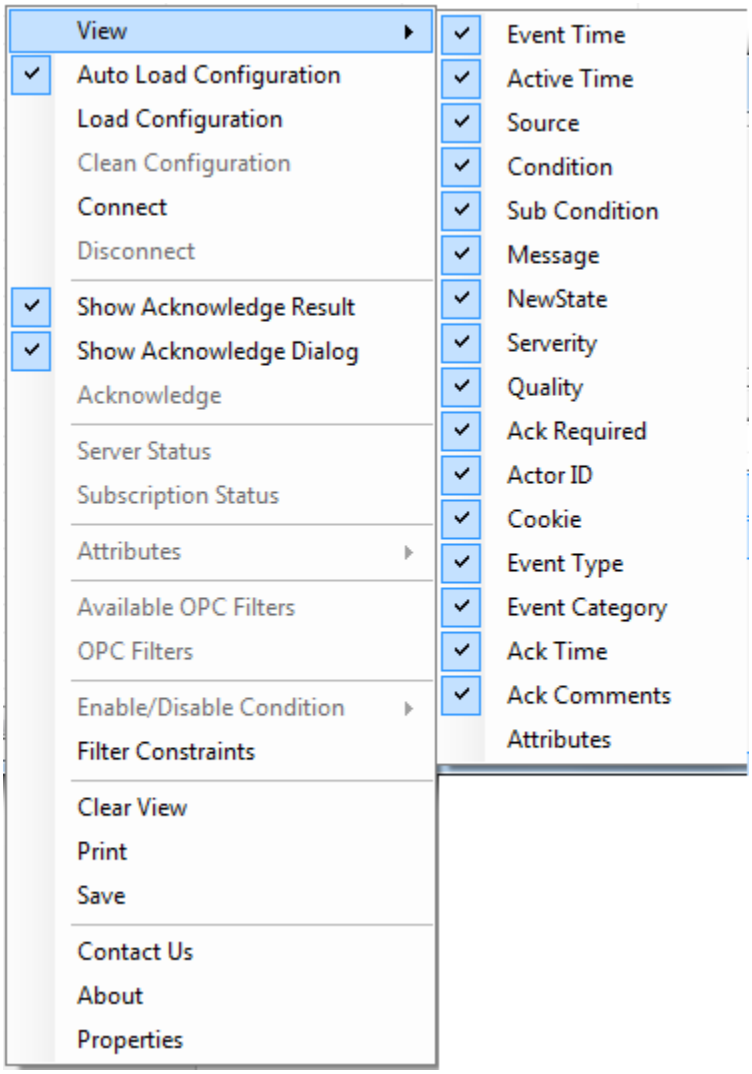


Figure 54: View Menu Item

4.4. Auto Load Configuration

In order to load a configuration automatically in the application startup you need to check the **Auto Load Configuration** item from the popup menu then restart the application.

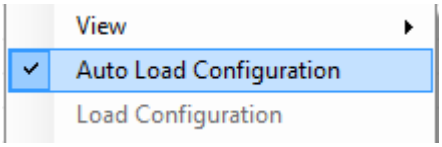


Figure 55: Auto Load Configuration Menu Item

4.5. Load Configuration

In order to load the latest running configuration you should click on the **Load Configuration** item from the popup menu.

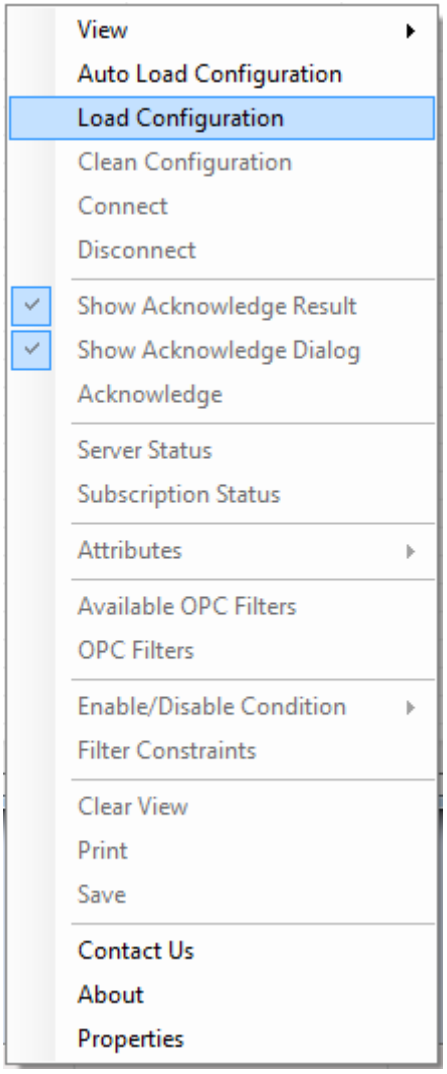


Figure 56: Load Configuration Menu Item

4.1. Clean Configuration

You can clean the configuration by clicking on the **Clean Configuration** item from the popup menu.

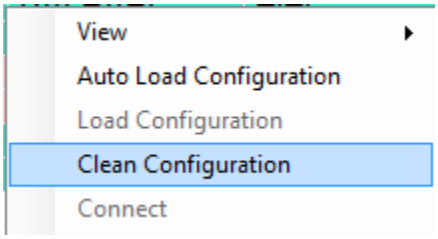


Figure 57: Clean Configuration Menu Item

4.2. Connect to an AE OPC Server

To connect to an AE Server choose the menu item **Connect** from the popup menu. The following window will be displayed.

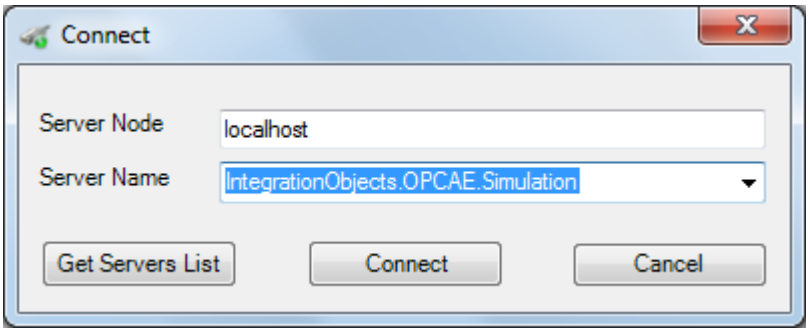


Figure 58: Connect From Context Menu

In order to connect to an OPC AE Server, you need to:

- Enter the server node of the OPC AE Server,
- Click on **Get Servers List** button to get all the available OPC AE Servers in the selected node, Select the ProgID AE Server from the returned server list then
- Click on the **Connect** button.

4.3. Disconnect from an AE OPC Server

If your application is connected to an OPC AE Server and you want to disconnect from it, click **Disconnect** from the popup menu.

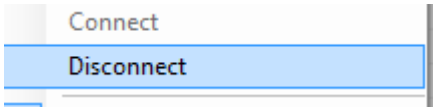


Figure 59: Disconnect Menu Item

4.4. Acknowledging alarms

To acknowledge an alarm requiring a response, the user may double click on the entry, or click the **Acknowledge** menu item from the context menu.

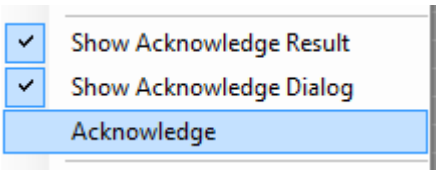


Figure 60: Acknowledge Menu Item

When the user right clicks on an alarm that requires acknowledgment and chooses Acknowledge, the following dialog box will appear.

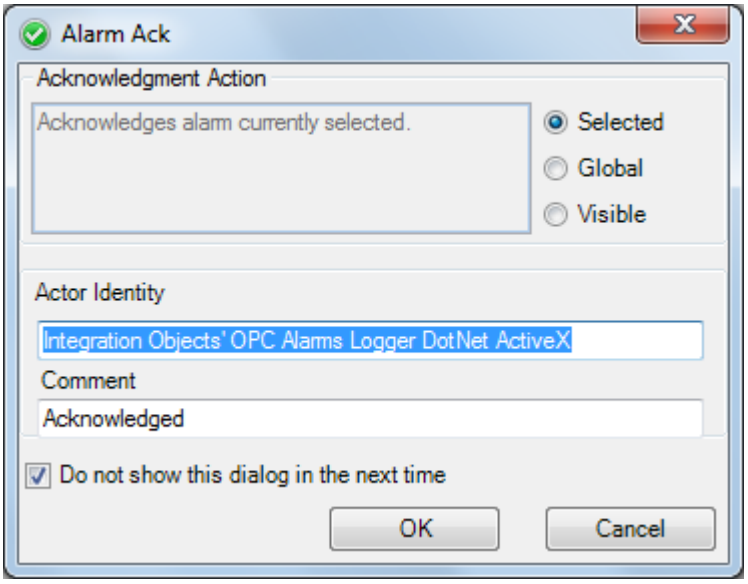


Figure 61: Acknowledging Alarms

In the Alarm Acknowledgment dialog box, users have the possibility to choose between three acknowledgment options which are listed below:

- **Selected**: Acknowledges alarm currently selected.
- **Global**: Acknowledges all alarms received. This option quickly acknowledges all alarms from the current view.
- **Visible**: Acknowledges all visible alarms. For example, if the size of the Viewer shows five alarms, and a total of eight alarms came in, only the five visible alarms are acknowledged

Users can also add comments and change the actor identity. If the user wants to always have the same acknowledgment options, he can click the *Do not show this dialog the next time* check box.

To display the acknowledgment options dialog once again, he may mark the **Show Acknowledge Dialog** from the popup menu.

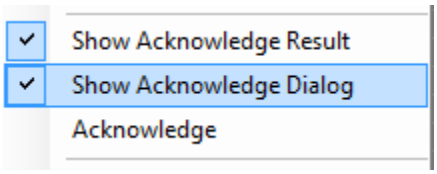


Figure 62: Show Acknowledge Dialog Menu Item

To display the acknowledgment result message box in case the operation failed, he may mark the **Show Acknowledge Result** from the popup menu.

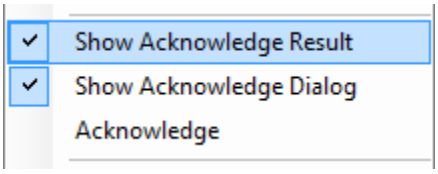


Figure 63: Show Acknowledge Result Menu Item

4.5. Server Information

To view the OPC server status, choose the **Server Status** menu item in the popup menu and the server status dialog box will appear. This window shows all static information about the AE Server.

The server status includes:

- Server start time
- Current time
- Time of last update sent
- Current OPC server state
- Major version of server
- Minor version of server
- Build number of the server
- Vendor specific information

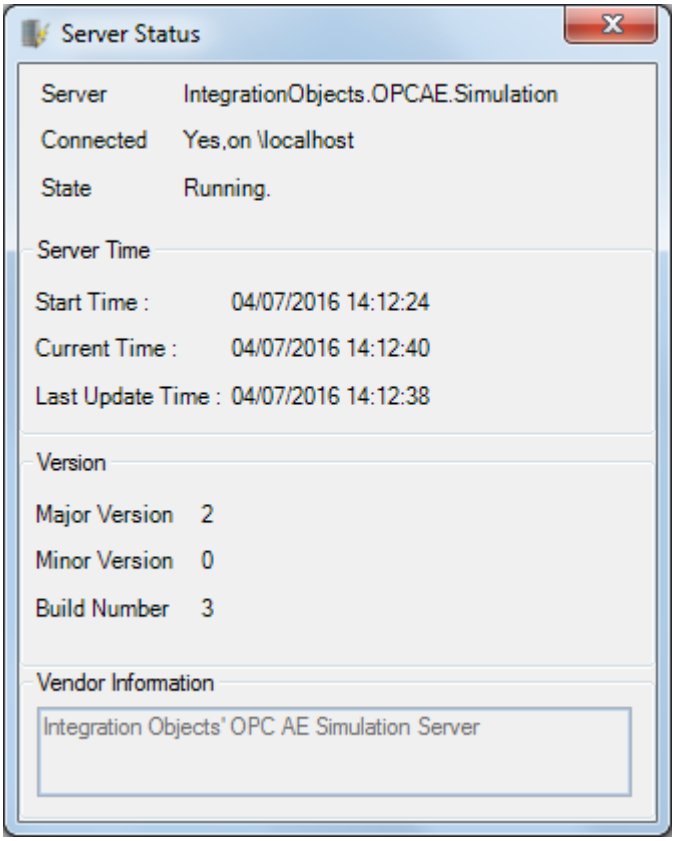


Figure 64: Server Status

4.6. Event Subscription Status

To view the Event Subscription status, choose the **Subscription Status** menu item in the context menu and the following dialog is displayed. This dialog allows users to get subscription state information and to modify the subscription state.

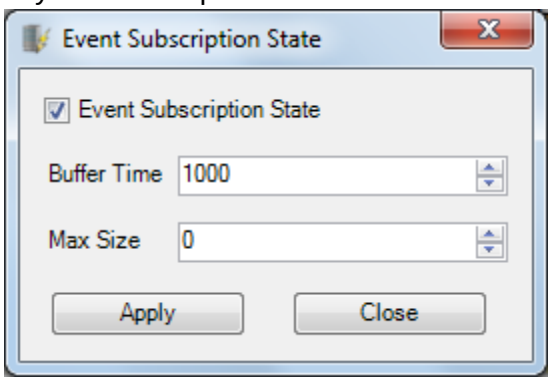


Figure 65: Event Subscription State

The event subscription state is presented as a check box. When checked, the event subscription is active and it sends events notifications.

The buffer time is the time in milliseconds that the server can hold back the notification to buffer multiple events. Example, 1000 would indicate that the server buffer events along 1 second and sends the whole set of events as one callback.
The max size is the maximum number of events to buffer before sending events.

4.7. Select Returned Attributes

To retrieve the attributes of an existing Event Subscription, the user should select the **Select Returned Attributes** menu item. For each event category, Select Returned Attributes picks out the attributes to return. This method can be called many times in order to specify the attributes to return for each unique event type and event category pair. If this is called multiple times for the same event type and event category pair, then the latest call will be considered.

If you choose the **Select All Attributes for All Categories**, all server event attributes will be displayed.

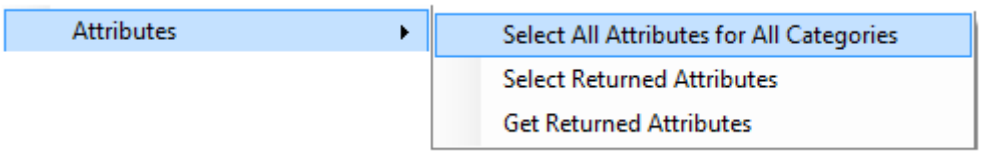


Figure 66: Select All Attributes for All Categories Menu Item

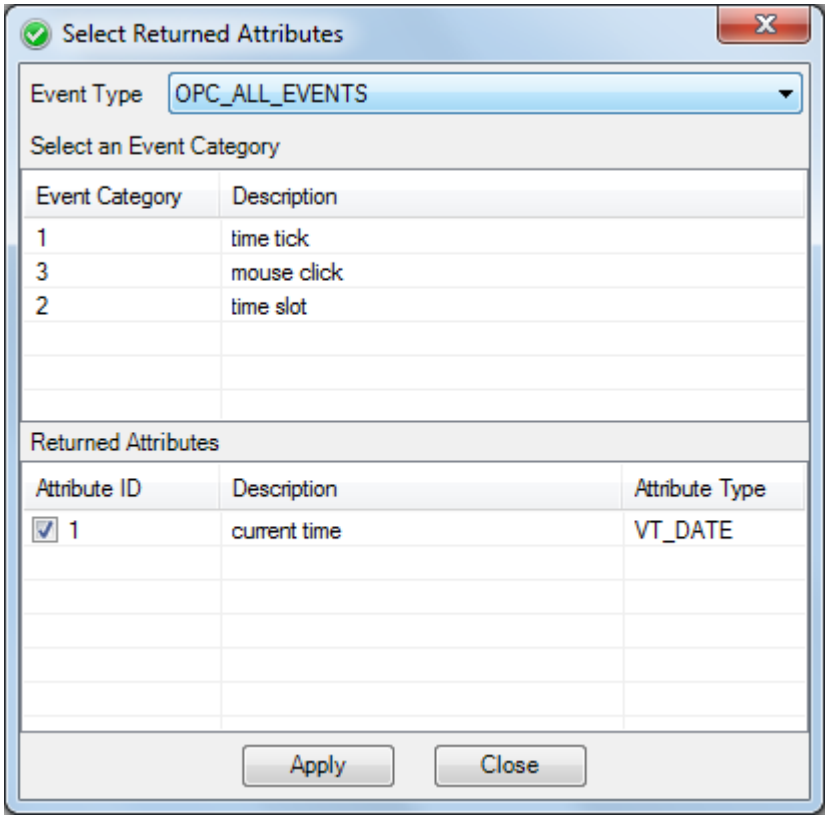


Figure 67: Select Specific Server Returned Attributes

4.8. Retrieve Returned Attributes

To get the attributes, the user should click the **Get Returned Attributes** menu item. For each event category, the following dialog retrieves the attributes previously specified by the user in the Select Returned Attributes dialog.

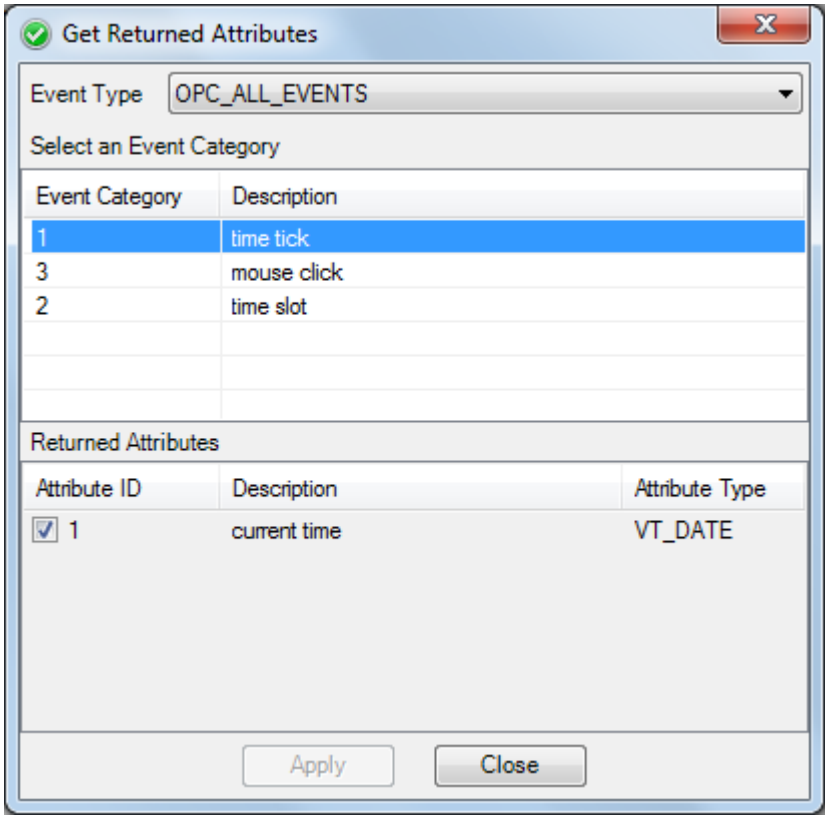


Figure 68: Get Specific Server Returned Attributes

4.9. Enable/Disable condition by areas and sources

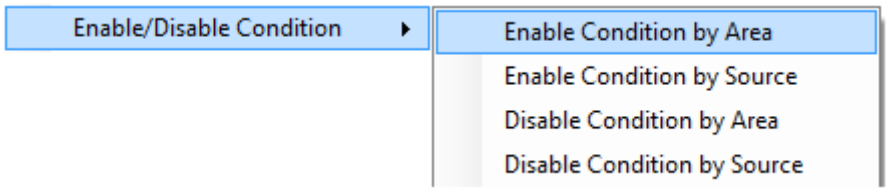


Figure 69: Enable/Disable Condition Menu Item

- **Enable condition by area**: Enables condition alarm events in an area.
- **Enable condition by source**: Enables condition alarm events for a given source.
- **Disable condition by area**: Disables condition alarm events in an area.
- **Disable condition by source**: Disable condition alarm events for a given source.

The dialog box below is the “enable condition by source.” The user should navigate into the tree and check the wanted sources names in order to enable conditions for a given source.

The similar thing should be done if the user wants to enable a condition by area or to disable a condition by source or by area.

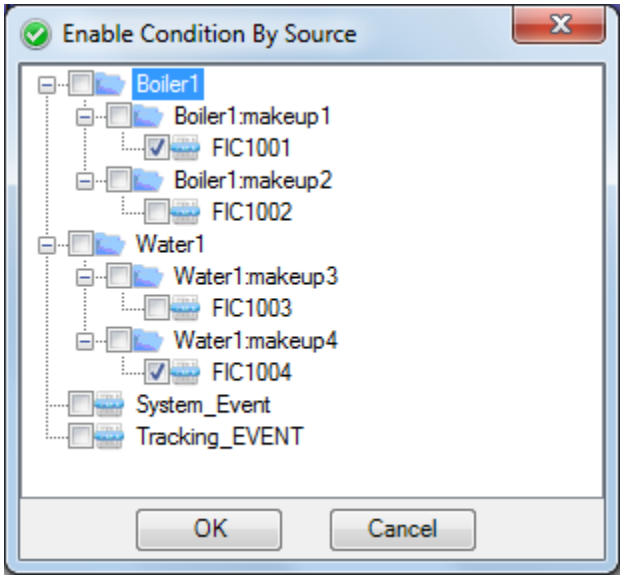


Figure 70: Enable Condition by Source

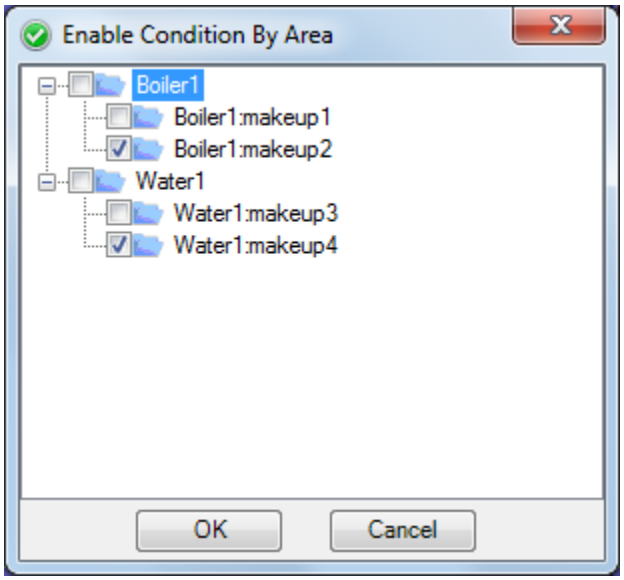


Figure 71: Enable Condition by Area

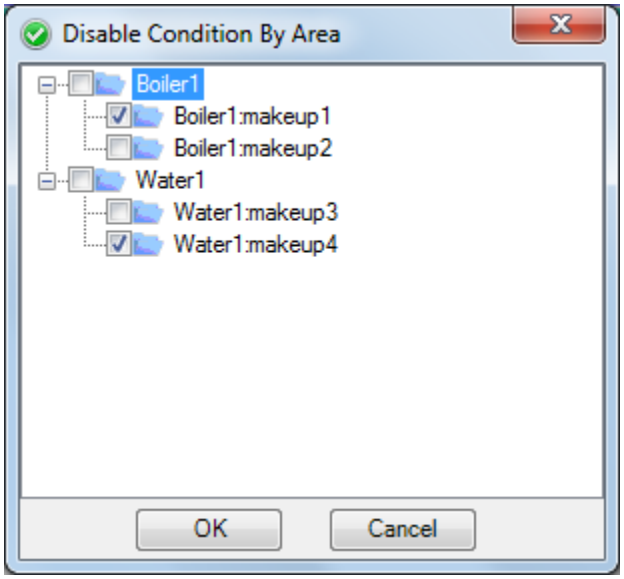


Figure 72: Disable Condition by Area

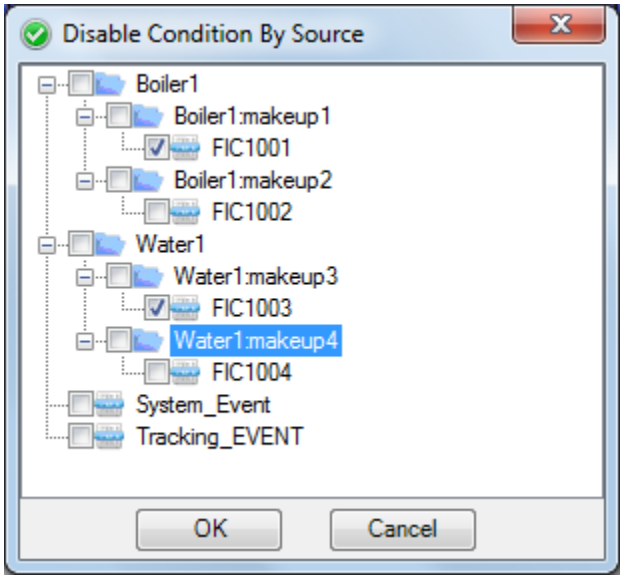


Figure 73: Disable Condition by Source

4.10. Available filters

To view the filters of an OPC alarms and events server, the user should select the **Available OPC Filters** menu item and the following dialog screen appears:

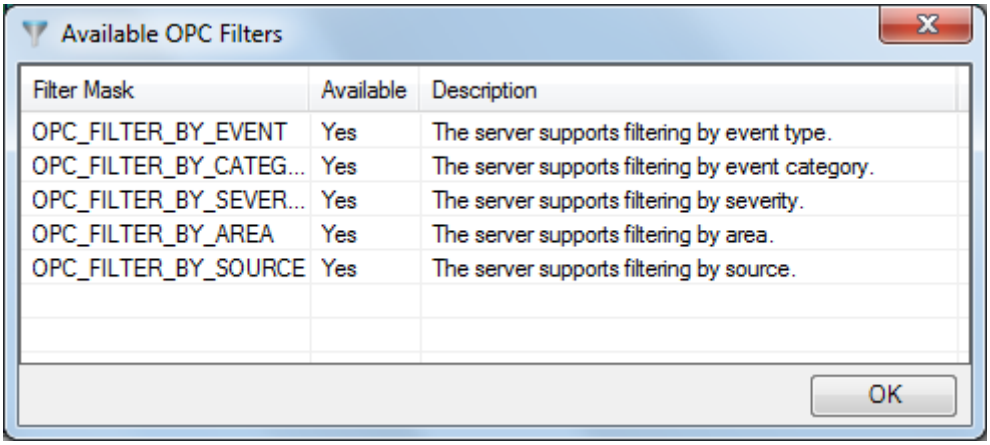


Figure 74: Available Filters

4.11. OPC Filtering Configuration

To define OPC Filters, click the **OPC Filters** menu item from the context menu. The OPC Filtering dialog is used to set criteria for including or excluding specific alarms in the control. Users can specify the event types, severity, event category, areas or sources to be filtered.

The user can set filters on any event subscription in order to limit the events that he will be notified of. To setup a filter for an event subscription, the user can use the following criteria:

- **Filtering by event type**: only events satisfying the criterion “Event Type” will be returned.
- **Filtering by event categories**: only events satisfying the criterion “Event Categories” will be returned.
- **Filtering by areas and sources**: only events satisfying the criterion “Existing in these areas or having these sources” will be returned.
- **Filtering by Severity**: only events satisfying the criterion “Events that have a severity between the min and the max severity” will be returned.

Users can select multiple criteria; they will be logically added together. This will cause all events satisfying these selected criteria to be returned.

If the user wants to get only specific event alarms, he may select the wanted check boxes.

The list of the event categories on the left indicates the categories to be filtered. The events on the right are the available categories. Select an event category and click on buttons (>> and <<) to move categories from one area to the other.

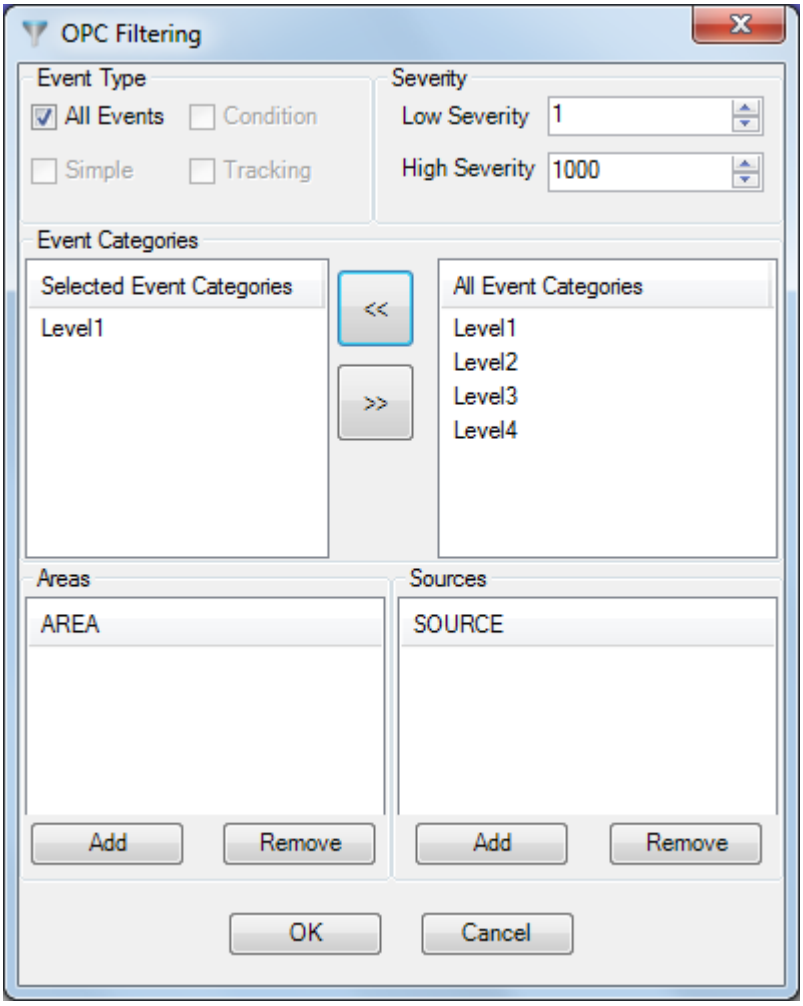


Figure 75: Configure OPC Filtering Alarms

The list of areas indicates the areas containing the source names needing to be filtered. To add a new area or a new source, click the **Add** button and a dialog box appears. The user may remove an area or a source by clicking the **Remove** button.

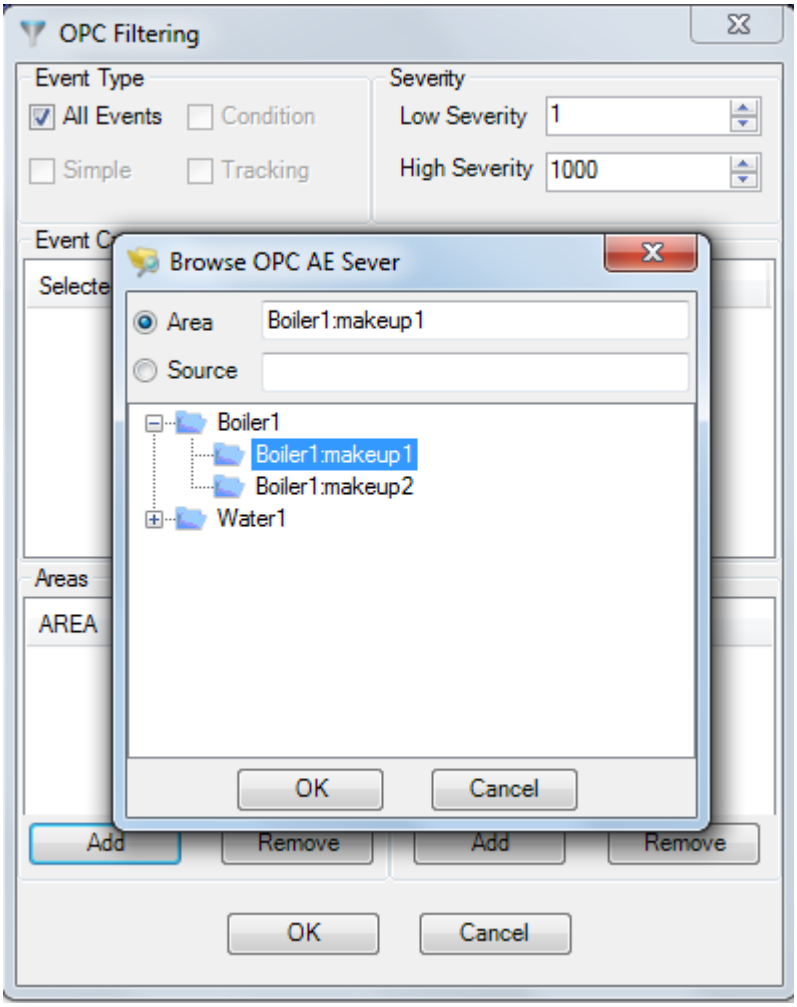


Figure 76: Add Areas

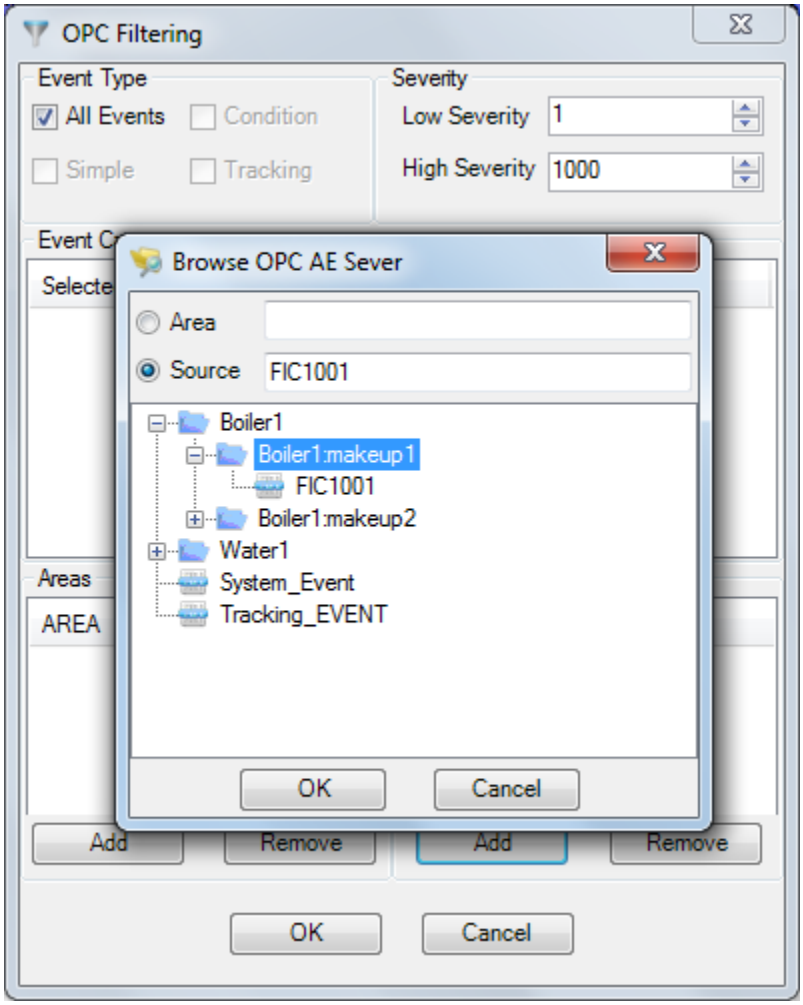


Figure 77: Add Sources

4.12. Apply Filter Constraints

By default, the AE Logger is configured to display all incoming alarms. To customize which alarms are displayed, right-click on the control, select **Filter Constraints** and build an equation to restrict only the alarms that respond to the build filter constraint.

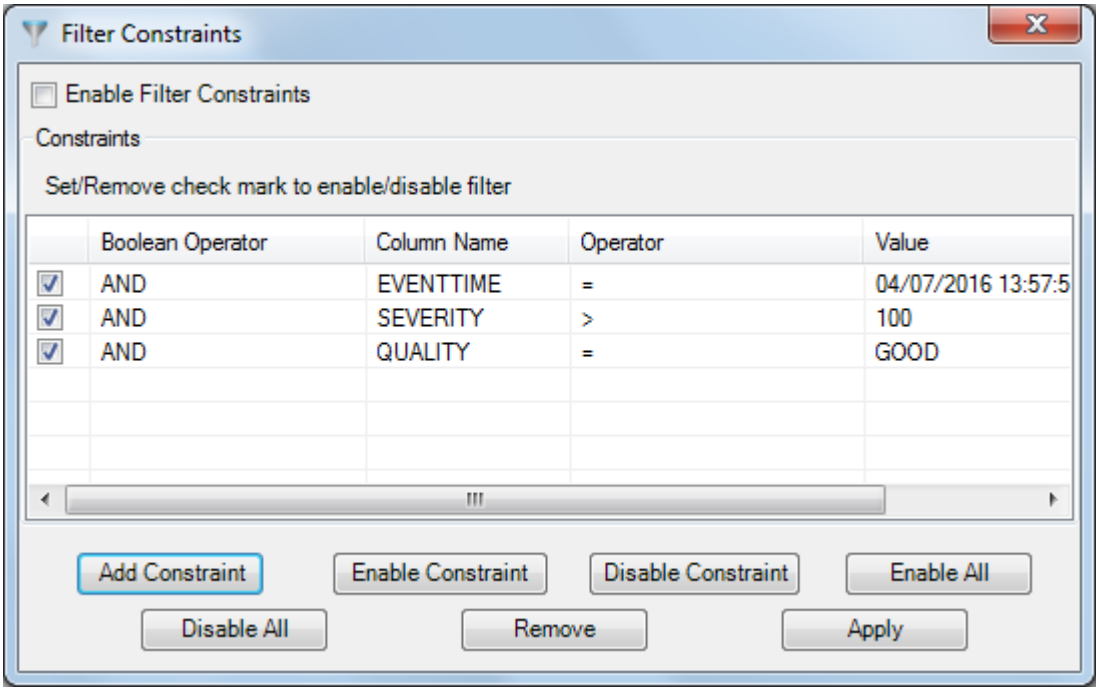


Figure 78: Filter Constraints Window

4.13. Clear View

The end user may clear the list view by clicking on the **Clear View** item in the popup menu as shown in the below figure.

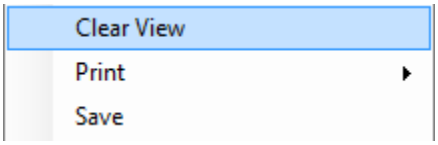


Figure 79: Clear View Menu Item

4.14. Print

The end user can print the available alarms in the list view by clicking on the **Printer** item of the popup menu.

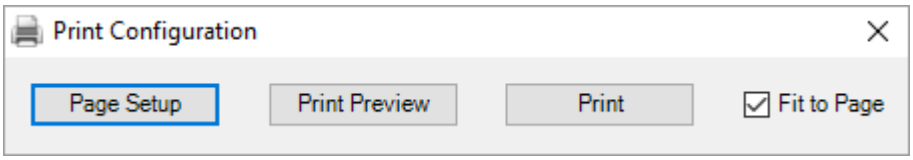


Figure 80: Print Configuration

The Print Configuration dialog box shown above allows the following actions:

- Setup the page by clicking on the **Page Setup** button and choosing either the portrait or the landscape view

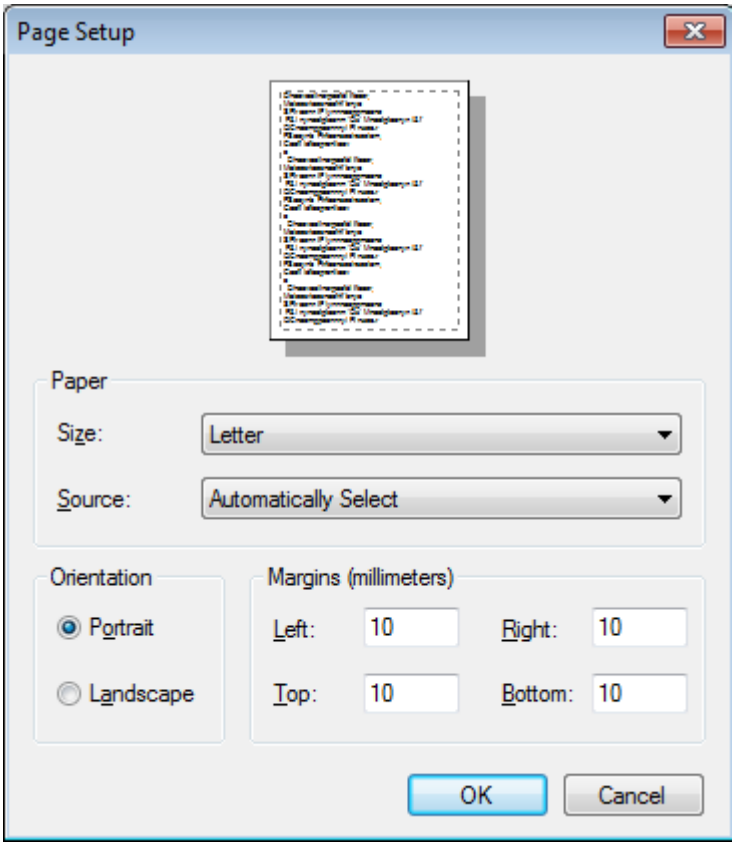


Figure 81: Page Setup Dialog Box

- Preview the print before printing by clicking on the **Print Preview** button:

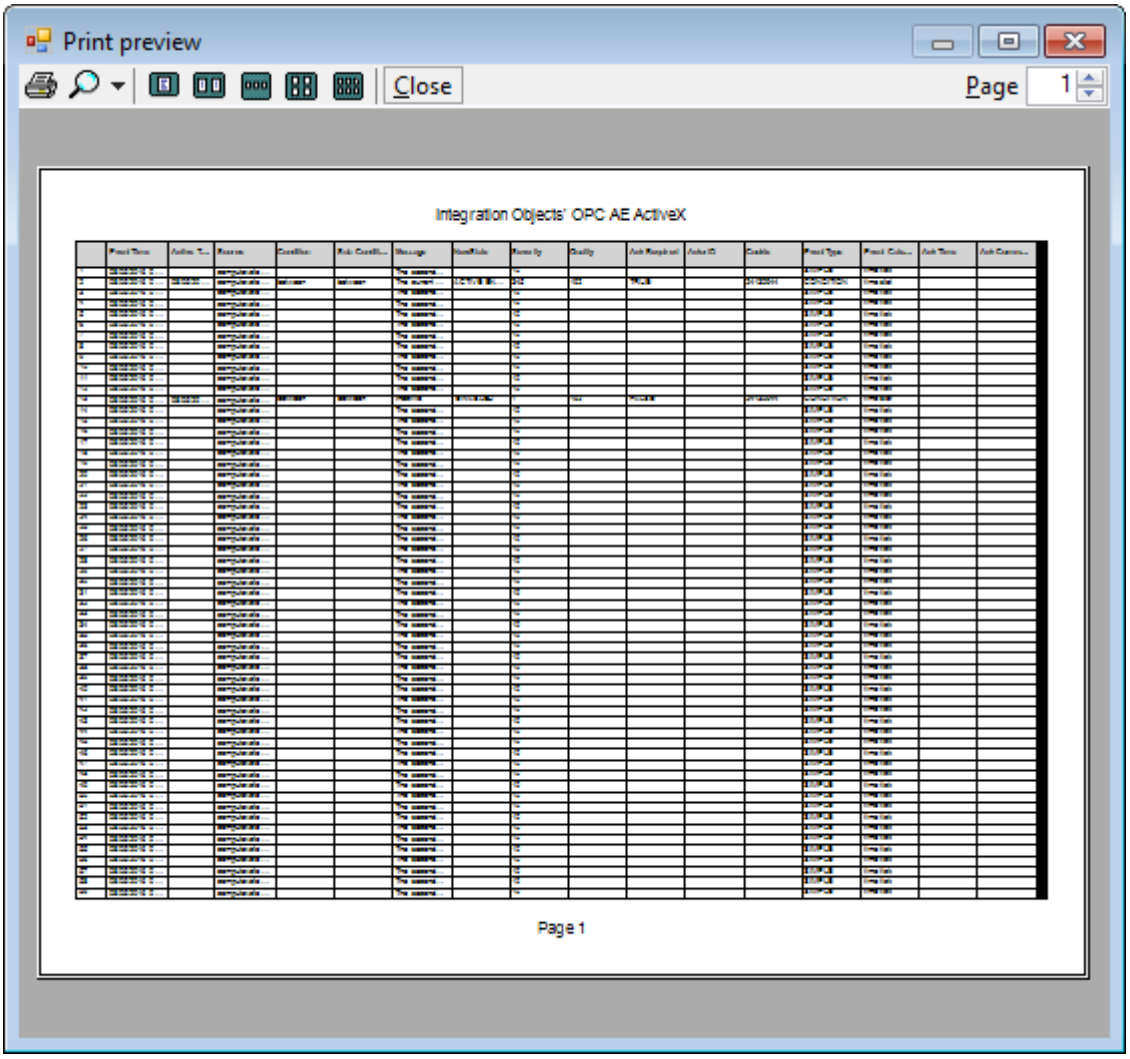


Figure 82: Preview Print Dialog Box

- Make the printing fits the page by checking the Fit to Page check box

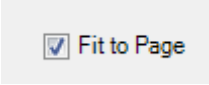


Figure 83: Fit to Page Check Box

- Choose the printer and print the alarms by clicking on the Print button

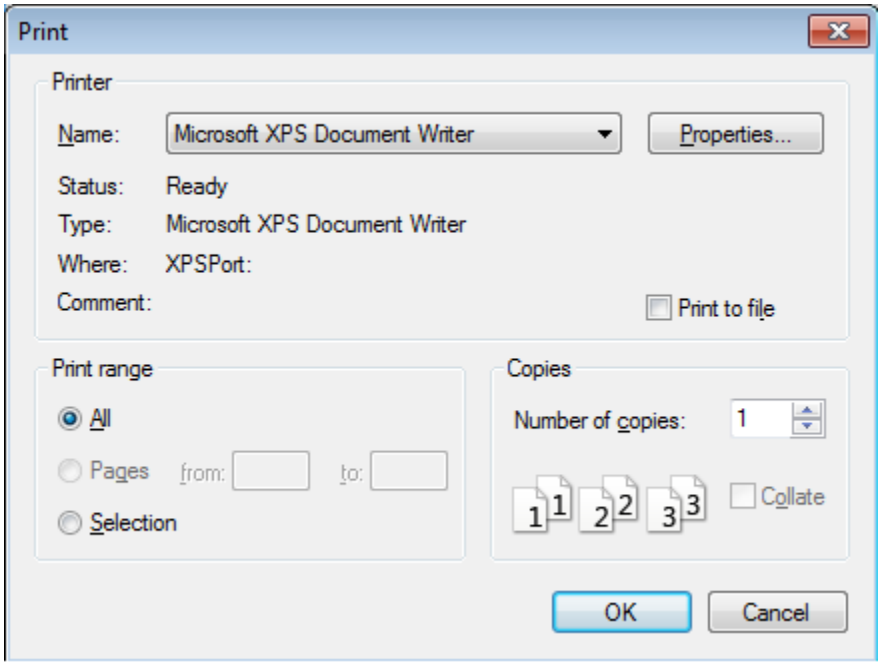


Figure 84: Choose Printer

4.15. Save

The end user can save the alarms in a csv file by clicking on the **Save** item of the popup menu.

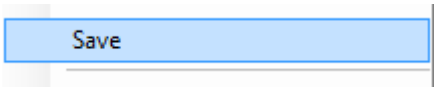


Figure 85: Save Item Menu

4.16. Properties

To view or modify the .Net Alarm Logger properties, select the **Properties** item in the popup menu.

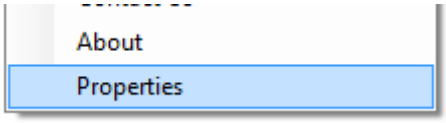


Figure 86: Properties Menu Item

4.16.1. OPC Options Configuration

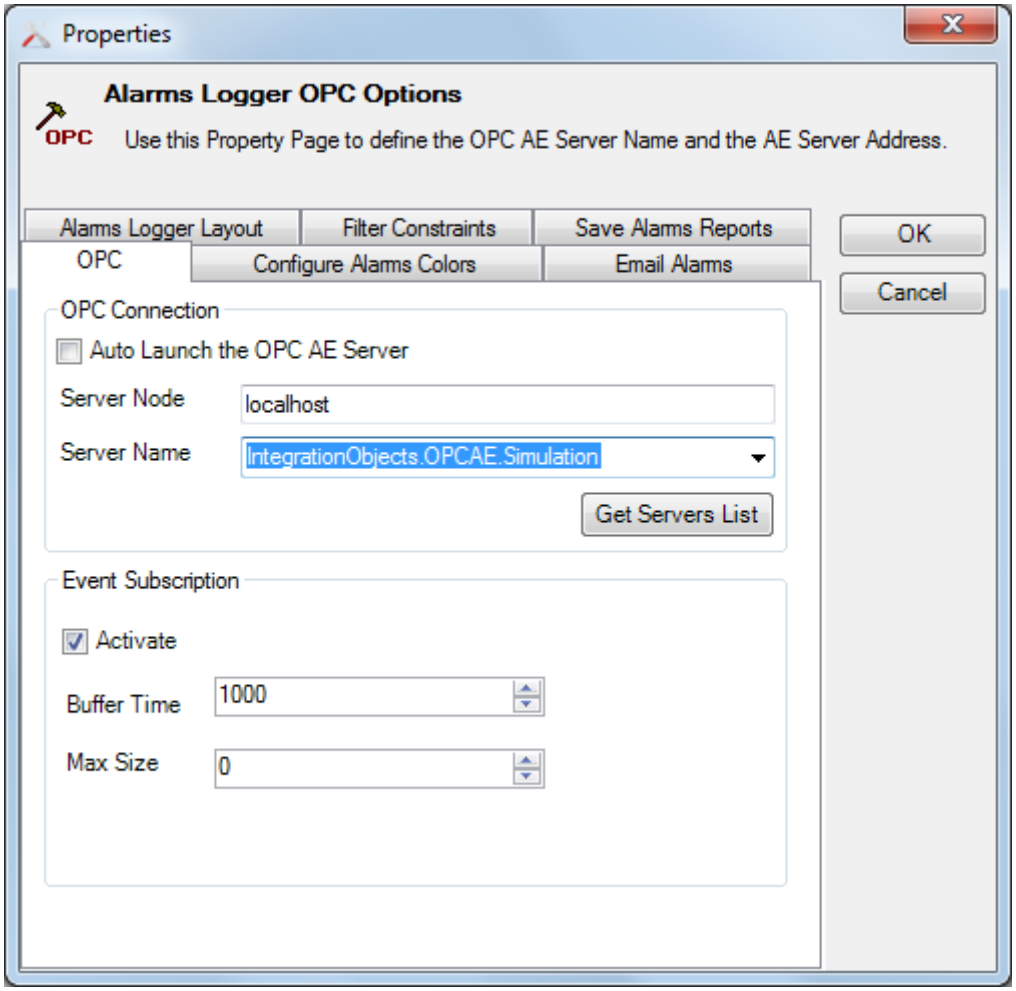


Figure 87: Alarms Logger OPC Options

The above figure represents the first tab sheet of the properties dialog, which is used to define the OPC Connection and Event subscription parameters.

- *OPC Connection*: To pick up an AE server, you should enter the server node, click on the **Get Servers List** button then select the server ProgID from the returned OPC Server List.
- *Event Subscription*: To set the event subscription parameter, you can either activate or deactivate the alarm and enter the buffer time and max size values.

4.16.2. OPC Alarm Color Configuration

Integration Objects' OPC AE Alarm Logger .Net ActiveX offers several options to configure GUI to be efficient and to respect the users' preferences. To change the configuration use the **Configure Alarms Colors** tab sheet in the main properties window.

The next tab sheet gives users the possibility to adjust background and text colors for alarms. Colors are an important piece of information for the alarm display.

You may also choose to enable blinking of active and unacknowledged alarms by clicking on the "enable blinking" check box.

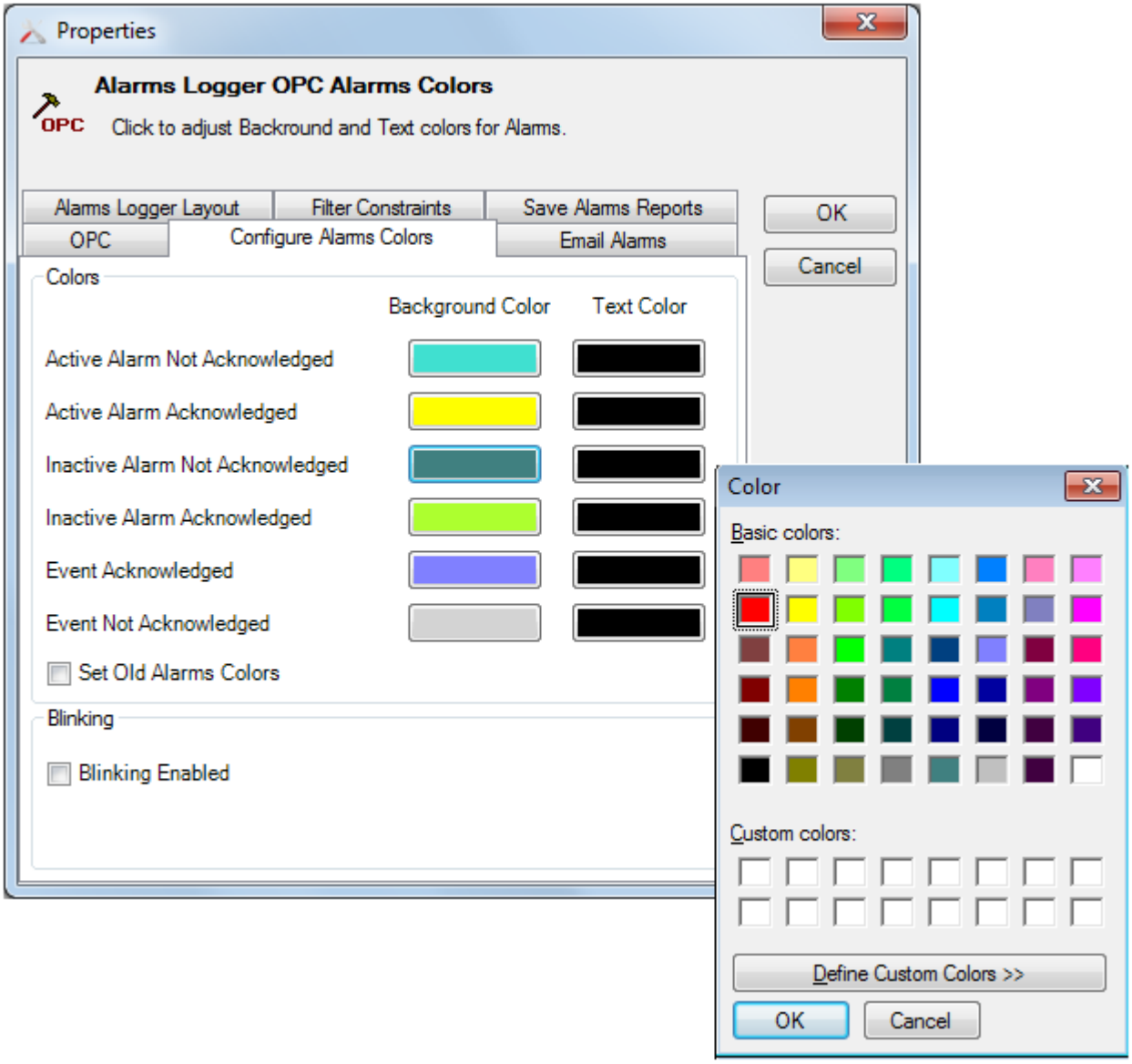


Figure 88: Configure Alarms Colors

4.16.3. Email Alarms Configuration

You can configure the application to send emails when an event or an alarm occurs. The user must provide the destination email address and SMTP server.

Integration Objects' OPC AE Alarm Logger .Net ActiveX has a tab sheet used to configure email notifications. When an alarm occurs, the application using the control automatically sends detailed notifications to the administrator by e-mail. The user can specify low or high severity for received alarms and events.

To enable mail notification, you should check the “enable mail notification” check box. If you choose to select the “send only critical alarms” option, you will receive only the incoming alarms that meet the severity level you define.

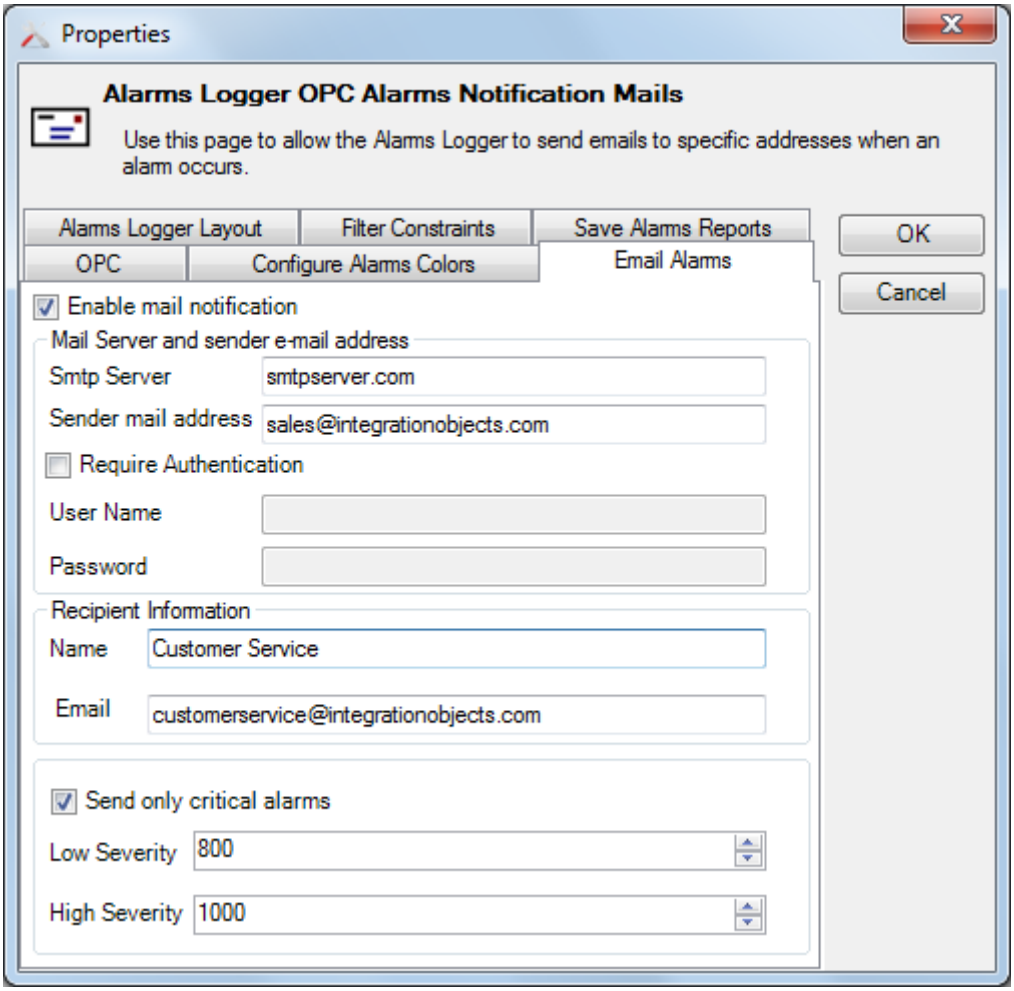


Figure 89: Configure Email Notifications

4.16.4. Control Layout Configuration

The Control layout tab is used to select the columns you want to appear on your control panel. The following dialog box allows users to view custom information, to pick up columns you want to display in the control panel, and to choose displayed format data to access further information.

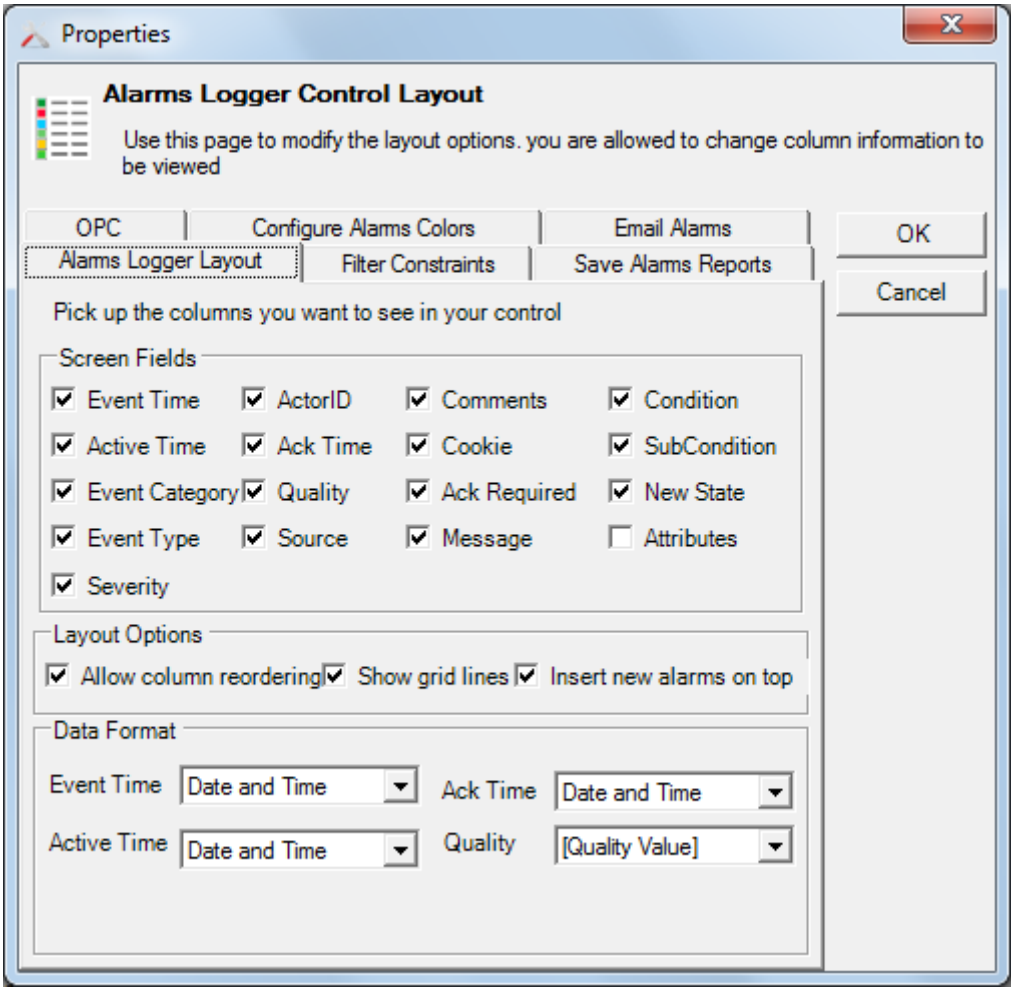


Figure 90: Alarms Logger Layout

This ActiveX control includes the following columns:

Column Heading	Description
Event Time	The time that the event was generated.
Active Time	The time that the event was active.
Source	The source item that caused the event to occur.
Condition	The condition name that caused the event to occur.
Sub Condition	The detailed cause of the event.
Message	The alarm message.
Event type	OPC specific event type (simple, tracking, condition).
Event category	One of the OPC event categories.
Quality	An indicator to the reliability of the event.

Actor ID	The name of the user that acknowledged the event.
Ack Comment	The comments entered by the user when he acknowledges the event.
Ack required	Indicates if the event requires acknowledgment.
Severity	The severity of an alarm.
Ack Time	The time when the user acknowledged the alarm.
Cookie	
NewState	The type of alarm (active, acknowledged …).
Attributes	Vendor specific server attributes.

Table 5: The ActiveX Columns

If the user wants to pick up only some screen fields to display in the control, simply uncheck the unwanted columns.
All columns are checked by default except the “Attributes” column which the user may mark if so desired.

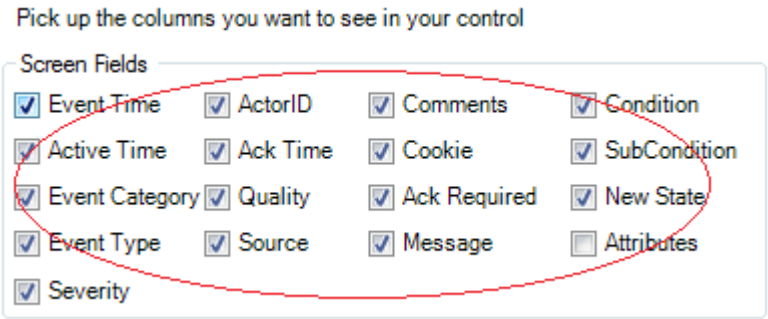


Figure 91:Pick Up Columns to Display

This Layout dialog allows users to define layout options for the control.
The user can choose the ability to drag-and-drop column headers to reorder columns in the control. By default, drag-and-drop reordering of columns is enabled (Figure 94).

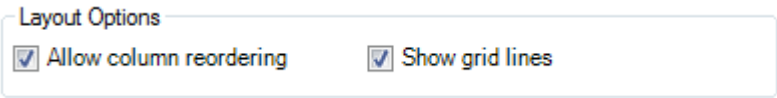
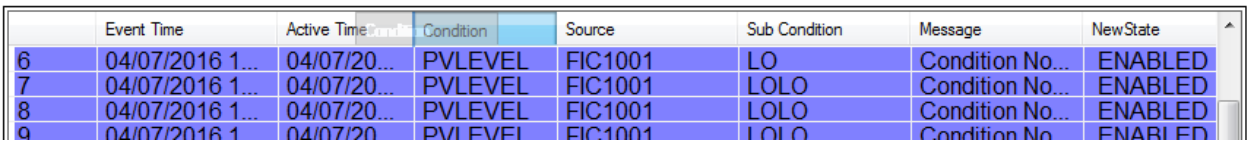


Figure 92: Enabling Reordering Columns

To move a column, click the column header once to select it. Then click and drag the column header to a new location (Figure 95).



	Event Time	Active Time	Condition	Source	Sub Condition	Message	NewState
6	04/07/2016 1...	04/07/20...	PVLEVEL	FIC1001	LO	Condition No...	ENABLED
7	04/07/2016 1...	04/07/20...	PVLEVEL	FIC1001	LOLO	Condition No...	ENABLED
8	04/07/2016 1...	04/07/20...	PVLEVEL	FIC1001	LOLO	Condition No...	ENABLED
9	04/07/2016 1	04/07/20	PVLEVEL	FIC1001	LOLO	Condition No	ENABLED

Figure 93: Drag-and-Drops Column Headers to Reorder Columns

We've decided to add a new preference on the layout page which allows you to select the time format for some screen field controls.

Along with choosing a custom date/time format, the user may choose to display date and time or just time for the three fields: **Event Time**, **Active Time** and **Ack Time**.

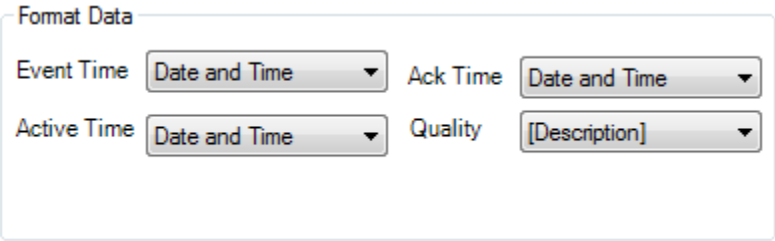


Figure 94: Customizing Data Format

4.16.5. Filter Constraints Configuration

The Filter constraints tab is used to set criteria for including or excluding specific alarms in the list view.

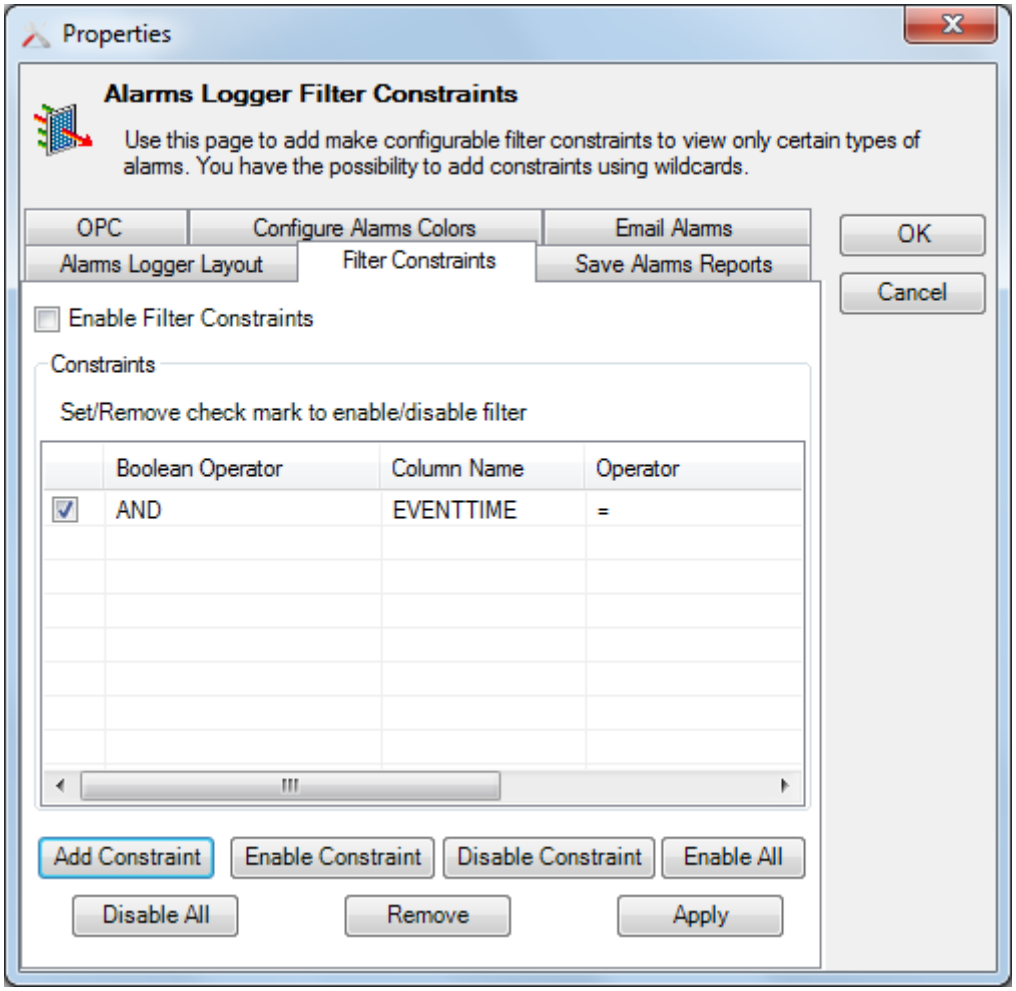


Figure 95: Filter Constraints

To define a filter constraint, click the **Add Constraint** button and the dialog (Figure 98) will appear. You must select the filter column name, the operator (=, #, <, …) and the value.

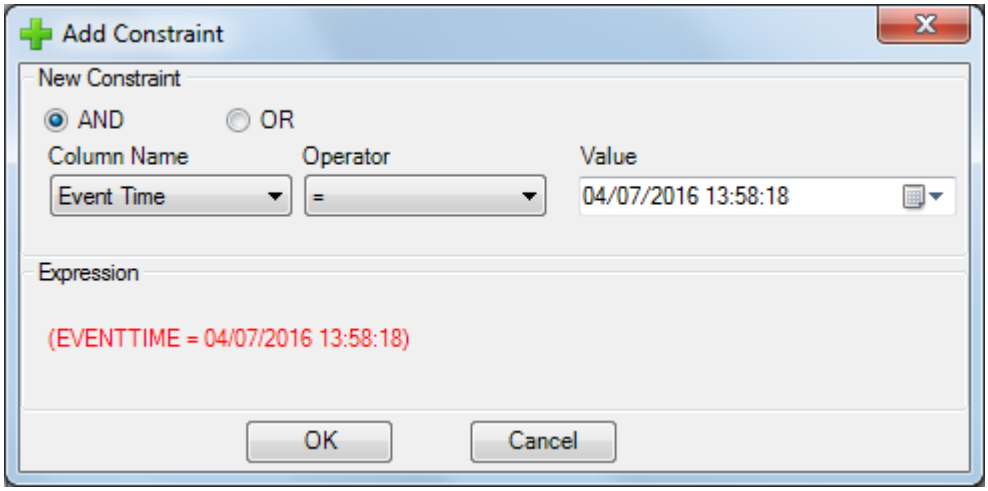


Figure 96: Add Constraint Dialog

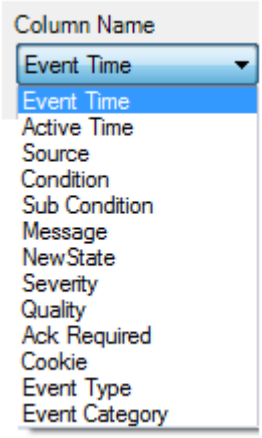


Figure 97: Select the Constraint Field

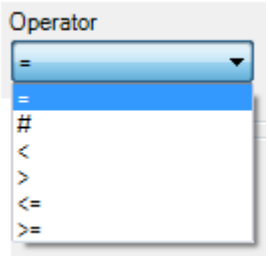


Figure 98: Select the Constraint Operator

Before moving to another tab, you should click the *Apply* button to save modifications. If you did not do so, the following message box will appear.

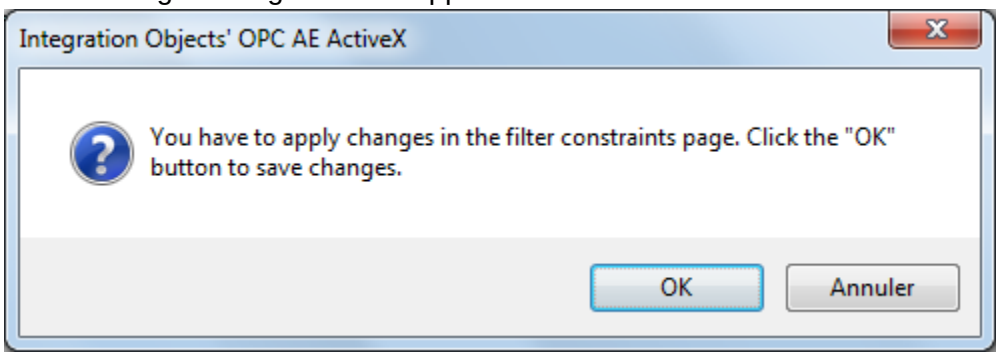


Figure 99: Warning Message Box

4.16.6. Save Alarm Reports Configuration

Integration Objects' OPC AE Alarm Logger .Net ActiveX allows the user to save the retrieved alarms into report files. To do that, the user has the possibility to define how frequently he wants the save to be performed (each minutes or after a certain number of minutes).

By default, alarms are saved each hour into the bin folder under the installation directory. If you are not pleased with this location, you can select another directory. Click **Browse Path** to select

the folder where you want to save the alarms. When saving the alarms report, the control creates a file in the chosen folder which is entitled:
“AlarmsReports{DAY]_{MONTH]_[YEAR]_{HOUR]_{MIN]_{SECONDS]” with the “csv” extension.

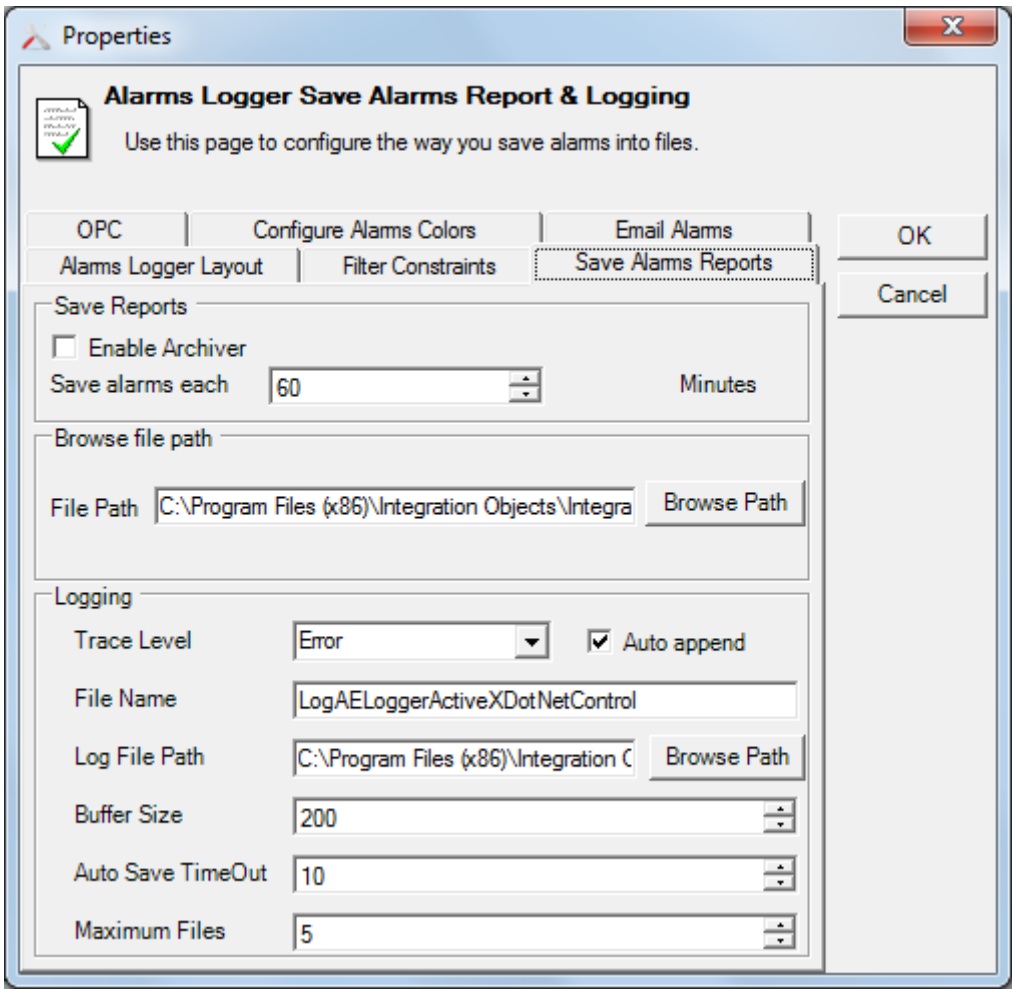


Figure 100: Save Alarms Reports

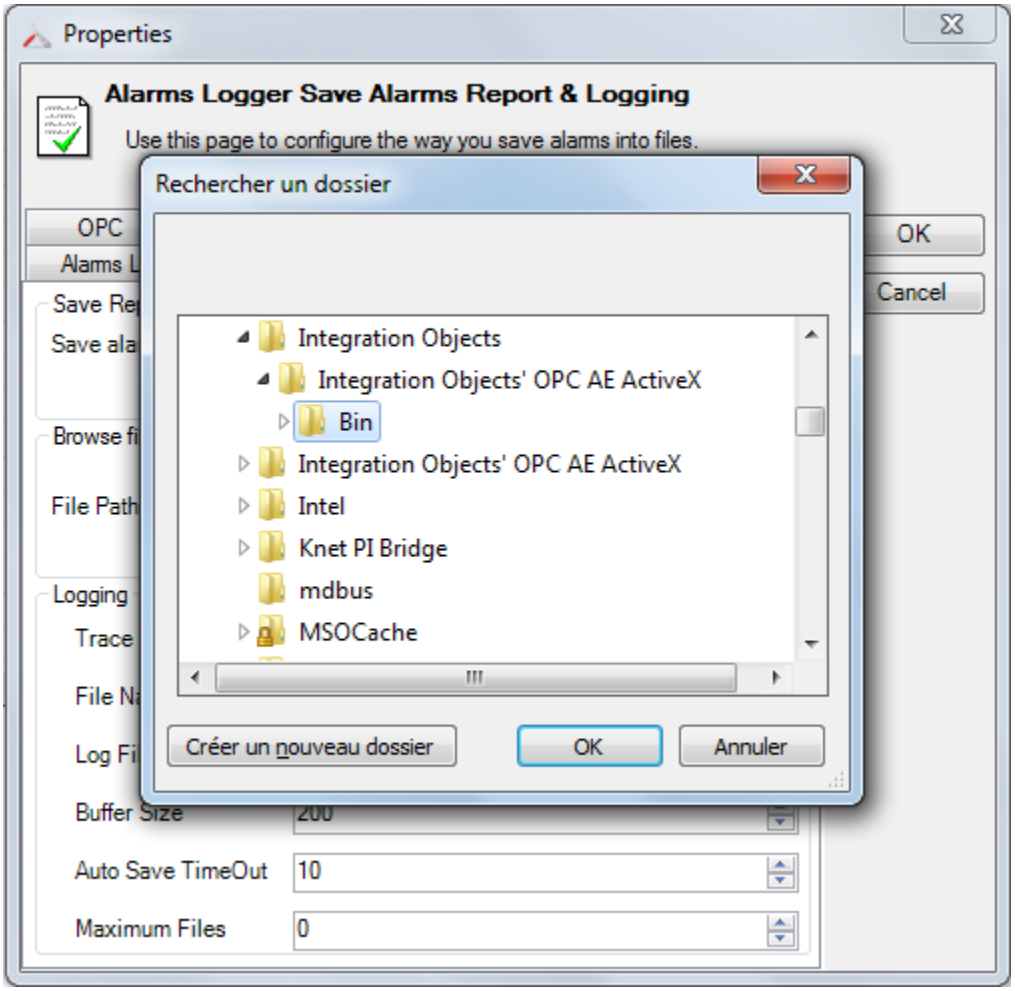


Figure 101: Select Reports Folder

4.17. Logging

Integration Objects' OPC AE Alarm Logger .Net ActiveX produces a log file named "LogAELoggerActiveXDotNetControl.LOG" that records errors and debugging information at runtime. If difficulties occur with the application, the log file can be extremely valuable for troubleshooting. When operations are running normally, the control will log very little information.

These log files are generated by default in the bin folder under the installation folder. These parameters all have default settings and can be changed by simply changing the information in the properties window.

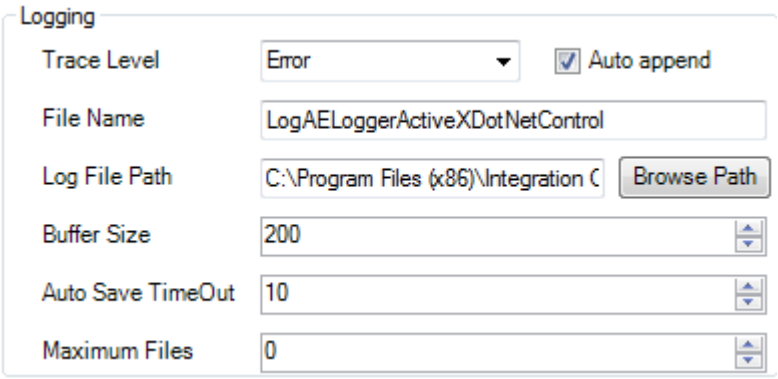


Figure 102: Log File Setting

Log Setting	Description	Default Value
Buffer Size	The maximum number of messages to be stored in the runtime memory before launching writes action in the hard disk. It must be greater than 100.	200
Trace Level	The type of log messages to be logged. The value can be Control, Error, Warning, Inform, and Debug.	Error
AutoAppend	Set to true to continue writing log messages in the existed log file or to false to create a new file.	TRUE
Log File Path	The path where the log file will be generated. It's set to the bin folder by default and can be modified by the user to change it to a customized one.	The Bin folder path
Auto SaveTimeout	Time to wait to read all messages from the buffer	10
Maximum Files	Maximum number of files	5

Table 6: Log File Properties

4.18. About Box Dialog

Click **About** from the menu to display the About Box for the control. The About Box contains the product name and version number as well as other information about the software and Integration Objects.

Figure 103: About Box

2. Registering the OPC AE Logger .NET ActiveX

In order to register the Integration Objects' OPC AE Logger .Net ActiveX, the end user needs to follow the steps below:

1. Open the Command Prompt window as administrator.
2. Locate the regasm.exe and IntegrationObjects.OPCAELoggerDotNetActiveX.dll already copied in your machine and copy the path
3. Type cd Path (**Path** is the path of the regasm.exe and the IntegrationObjects.OPCAELoggerDotNetActiveX.dll
4. Type **regasm IntegrationObjects.OPCAELoggerDotNetActiveX.dll /tlb:"C:\ IntegrationObjects.OPCAELoggerDotNetActiveX.tlb" /codebase**

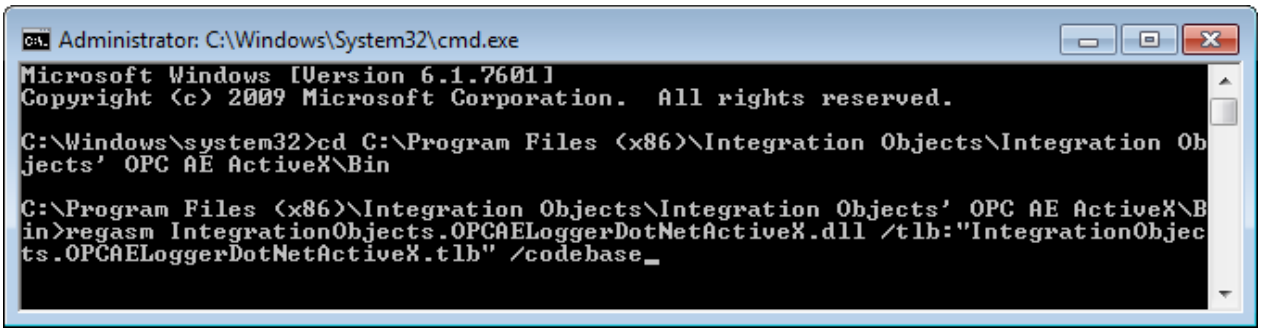


Figure 104: Registration Command Prompt

You can also use the **RegisterAELoggerDotNetActiveX** executable located in the bin folder under the installation directory. Once registered, you can add the **Activex.IOCSAELoggerActiveXCtrl** component in your workspace application.

3. Unregistering the OPC AE Logger .NET ActiveX

In order to unregister the Integration Objects' OPC AE Alarm Logger .Net ActiveX, the end user needs to follow the steps below:

5. Open the Command Prompt window as administrator.
6. Locate the regasm.exe and IntegrationObjects.OPCAELoggerDotNetActiveX.dll already copied in your machine and copy the path
7. Type cd Path (**Path** is the path of the regasm.exe and the IntegrationObjects.OPCAELoggerDotNetActiveX.dll
8. Type **regasm /u IntegrationObjects.OPCAELoggerDotNetActiveX.dll /tlb:"C:\ IntegrationObjects.OPCAELoggerDotNetActiveX.tlb" /codebase**

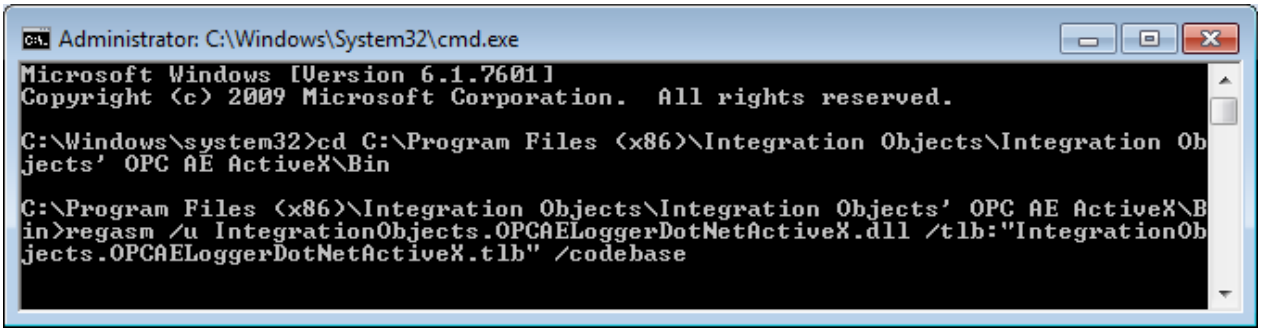


Figure 105: Un-registration Command Prompt

You can also use the **UnregisterAELoggerDotNetActiveX** executable located in the bin folder under the installation directory.

4. Deploying the .Net AE Logger in Microsoft Visual Basic 6.0

4.1. Create a Standard EXE

Run the Microsoft Visual Basic 6.0 and select a Standard EXE to create a new project as shown below.

Figure 106: Create a VB6 Standard EXE

4.2. Add the OPC AE Net Logger reference

1. Select the *Project* menu item and click on *References*

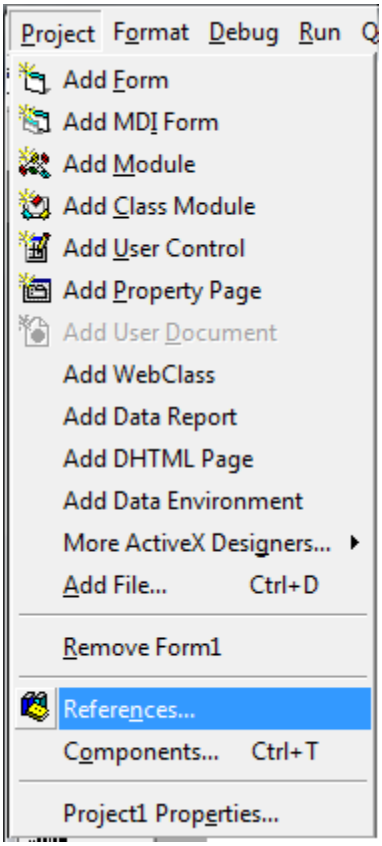


Figure 107: Select Projet Reference

2. Click on the Browse button and select the OPC AE Logger tlb file path (.:\Program Files (x86)\Integration Objects\Integration Objects' OPC AE ActiveX\Bin\DotNet\OPCAELoggerActiveX)

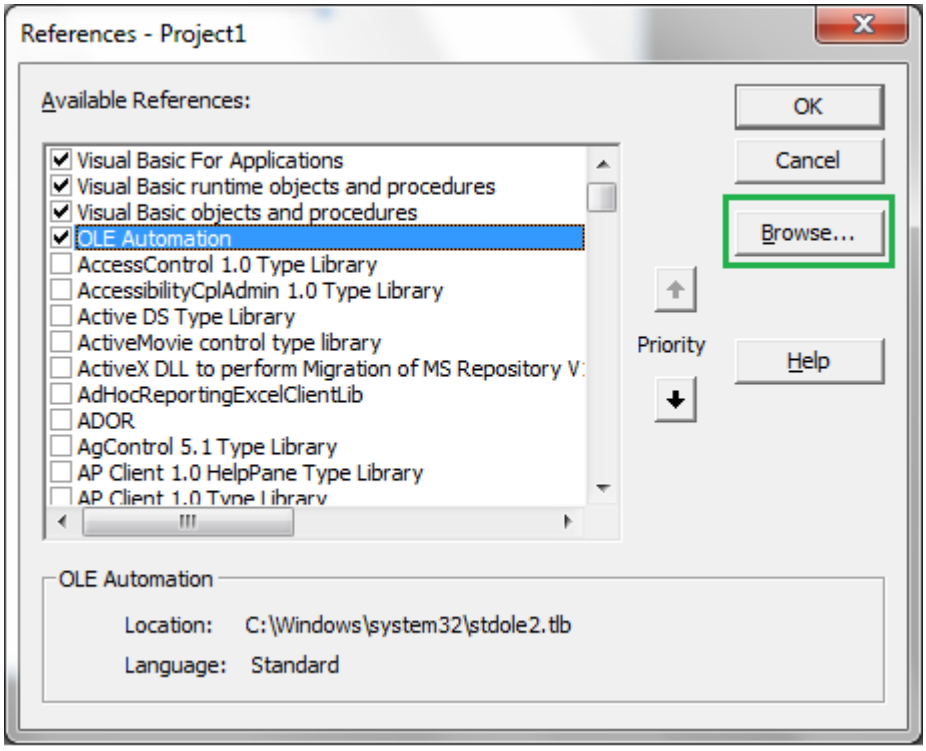


Figure 108: Browse the OPC AE Net Logger Path

3. Select the “IntegrationObjects.OPCAELoggerDotNetActiveX.tlb” and click on the *Open* button.

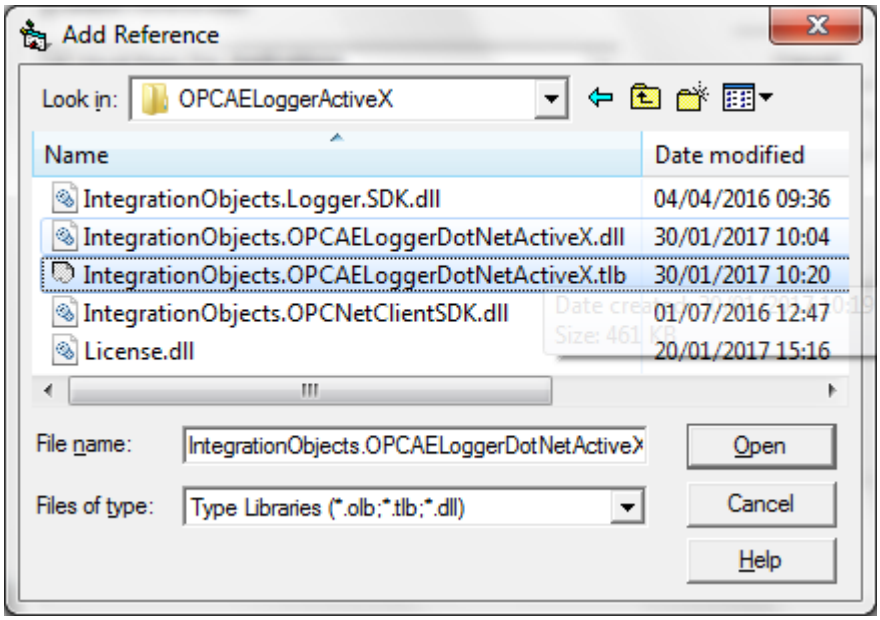


Figure 109: Select the type library (.tlb) file

4. Once the IntegrationObject_OPCAELoggerDotNetActiveX reference is checked, click on OK

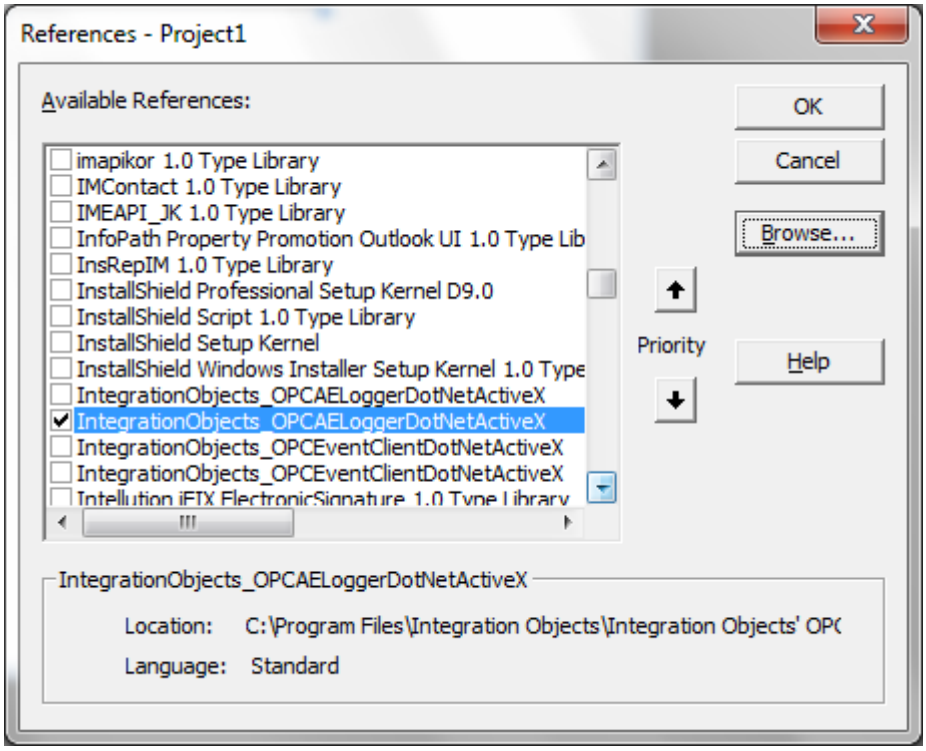


Figure 110: Check the OPC AE Net Logger Reference

4.1. Add the OPC AE Net Logger Component to the Toolbar

5. Select the *Project* menu item and click on *Components*

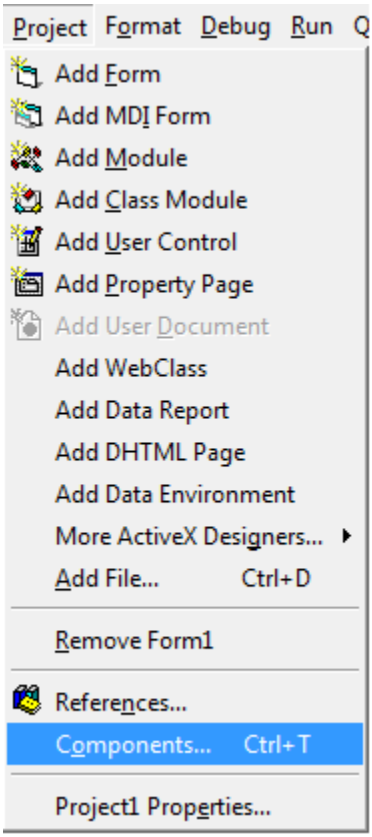


Figure 111: Select Projet Components

6. Check the IntegrationObject_OPCAELoggerDotNetActiveX component then click on the *OK* button

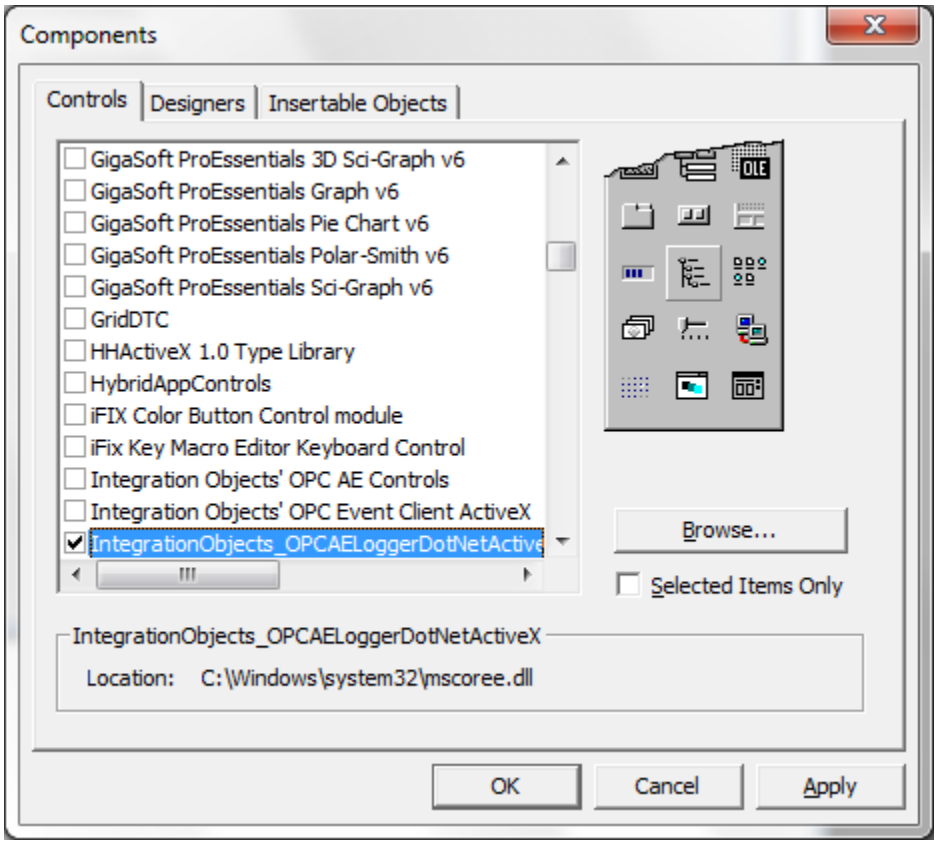


Figure 112: Select the OPC AE Net Logger Component

7. Select the IOCSAELoggerActiveXCtrl already added in the toolbar and draw the control in the form as shown below

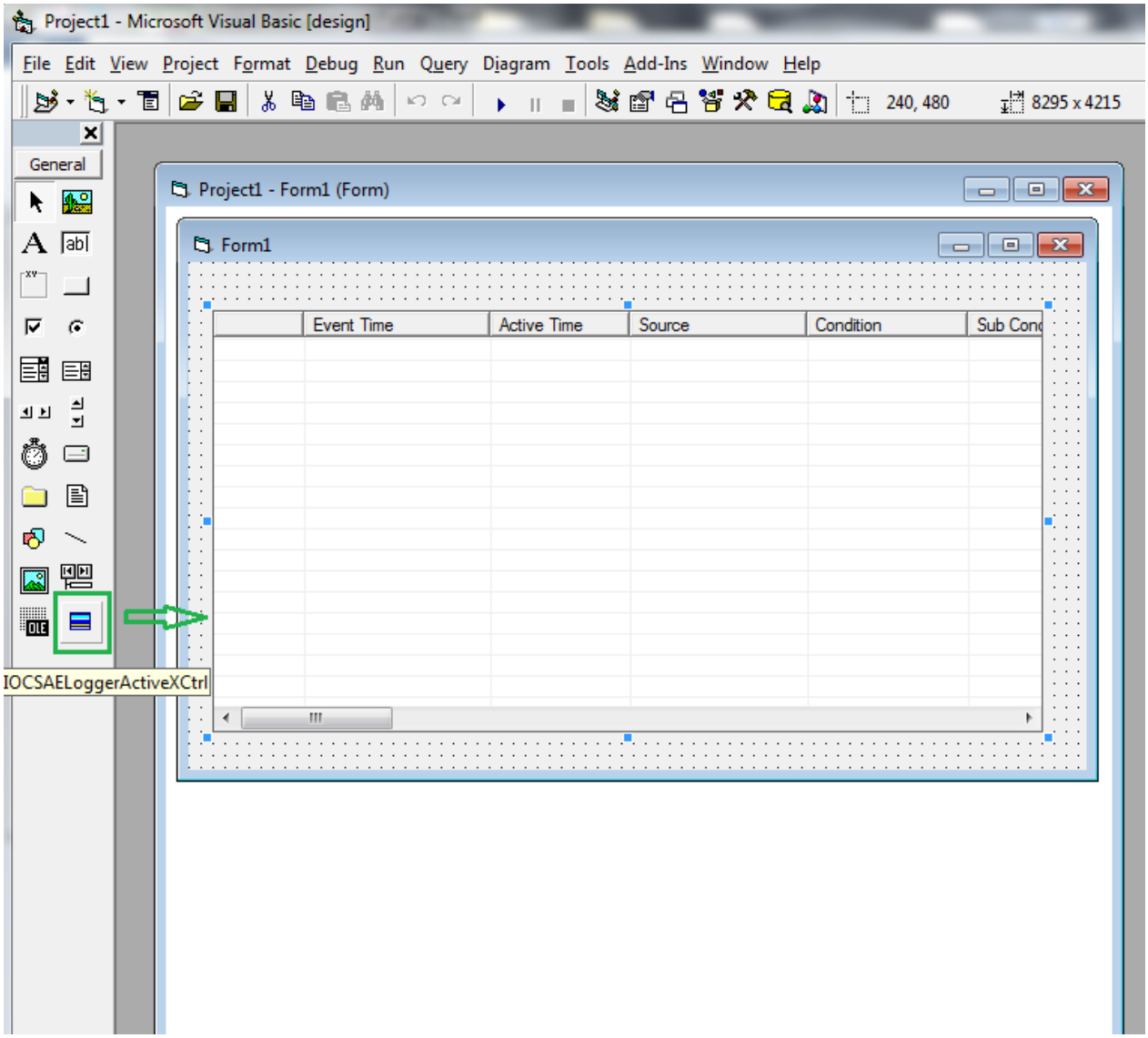


Figure 113: Add the OPC AE Logger Component to the Form

USING OPC EVENT CLIENT ACTIVEX

This control is a custom control which allows Visual Basic and other OLE Container applications to quickly and easily access data from any OPC Alarms & Events Server. The control's features enable the controlling application to establish a DCOM connection to an OPC A&E Server located on a local or on a remote machine. The application can subscribe to messages from the server and can be used to configure filters to receive only those of interest.
In this chapter, we will present the properties and methods of all objects in this Control.

1. IOEventClientCtrl

1.1. Logging

Integration Objects' Event Client produces a log file named "AELogEvent1.LOG" that records errors and debugging information. If difficulties occur with the application, the log file can be extremely valuable for troubleshooting. When operations are running normally, the control will log very little information.

This log file is generated by default in the bin folder under the installation folder. This path can be modified by the user using the "EventCltConfiguration.ini" configuration file incorporated by the toolkit which includes several logging parameters. These parameters all have default settings and can be changed at start-up by editing the configuration file, or by simply changing the information in the control properties window.

To change this file:

1. Open EventCltConfiguration.ini in a text editor.
2. Edit any of the parameters listed in the following tables:

Log Setting	Description	Default Value
LogFileMaxSize	The maximum log file size, in bytes. Once this size is reached during run-time, the log file is overwritten.	1048576*2 ~ 2 Mo (MegaByte)
TraceLevel	The trace level is a value telling what type of information to log: 0: Only errors messages are logged. 1: Some extra information. 2: Debugging information is logged. 3: Detailed debugging information. The higher the trace level, the more information is recorded. We recommend you to use level 0 for a better performance of the client application.	0
ArchiveLastLog	TRUE: Old file is copied to an intermediate file with incremental extension, before being overwritten. FALSE: Any pre-existing log file is erased and overwritten at start-up.	FALSE
LogFilePath	The path where the log file will be generated. It is set to the bin folder by default and can be modified by the user to change it to a customized one.	The Bin folder path

Table 7: Configuration File Properties

1.2. GetLocalOPCEventServers

GetLocalOPCEventServers	
Retrieve the list of local AE servers.	
listsrv = GetLocalOPCEventServers	
Return Value	**Description**
listsrv	An array containing the progIDs (names) of registered local OPC AE Servers.
Example	

```
Dim ServersList As Variant
ServersList = GetLocalOPCEventServers ()
```

1.3. GetOPCEventServers

GetLocalOPCEventServers	
Retrieve the list of local AE Servers or AE Servers located on a remote machine.	
listsrv = GetOPCEventServers(host)	
Argument	**Description**
host	The machine where the OPC Server is registered. Note that the host may be the IP address of the machine.
Return Value	**Description**
listsrv	An array containing the progIDs (names) of registered local OPC AE Servers.
Example	
Dim ServersList As Variant ServersList = GetOPCEventServers (“localhost”) Or ServersList = GetOPCEventServers (“\\IP_Address”) Or Dim HostName as String HostName = “\\IOServer” ServersList = GetOPCEventServers (HostName)	

1.4. CreateServer

CreateServer	
Add a new Server object to the IOEventClientCtrl ActiveX control.	
NewServer = CreateServer()	
Return Value	**Description**
NewServer	This method returns a pointer to the newly created IOEventServer object.

Example
Dim Server As New IOEventClientLib.IOEventServer Set Server = IOEventClient1.CreateServer ( )

1.5. Servers

Servers	
This method sends a reference to an IOEventServer object, given the server index.	
Server = Servers (index)	
Argument	**Description**
index	The server index.
Return Value	**Description**
Server	This method returns a pointer to the IOEventServer object given by the server index. If the server with the given index is not found, this method returns NULL. This method also returns NULL if the Evaluation period has expired.
Example	
Dim Server As IOEventClientLib.IOEventServer Dim Index as Variant Index = 1 Set Server = IOEventClient1.Servers (Index)	

1.6. AboutBox

AboutBox
Display the About Box for the control. The About Box contains the product name and version number as well as other information about the software and Integration Objects.
AboutBox()
Example
Call IOEventClient1.AboutBox ( )

2. IOEventServer

Property	Description
ComputerNode	[Read/Write]The machine where the OPC Server is registered. Note that the host may be the IP address of the machine.
ProgID	[Read/Write]The OPC Server name (ProgID).
ServerStatus	[ReadOnly]The current OPC AE Server Status Structure. This Structure is an **IOEventServerStatus** object.
CanFilterByEvent	[ReadOnly]A Boolean that defines the filter by Event.
CanFilterBySeverity	[ReadOnly] A Boolean that defines the filter by Severity.
CanFilterByCategory	[ReadOnly] A Boolean that defines the filter by Category.
CanFilterByArea	[ReadOnly] A Boolean that defines the filter by Area.
CanFilterBySource	[ReadOnly] A Boolean that defines the filter by Source.
CategoriesCount	[ReadOnly]Returns the number of categories defined within the A&E Server once the **QueryEventCategories** has been called.
ConditionsNumber	[ReadOnly]Returns the number of conditions related to the event category once **QueryConditionNames** has been called.
SubConditionsCount	[ReadOnly]Returns the number of subconditions related to the condition name once **QuerySubConditionNames** has been called.
SourceConditionsCount	[ReadOnly]Returns the number of conditions related to the concerned source name once **QuerySourceConditionNames** has been called.

2.1. ConnectToServer

ConnectToServer

Connect to the AE Server defined by *ProgID* and *ComputerNode* properties.	
LONG ConnectToServer (void) ; Public Function ConnectToServer( ) As Long	
Return Value	**Description**
hr	The operation result.
Example	

Server.ComputerNode = "\\IOServer"
Server.ProgID = "IntegrationObjects.AE.Simulation"
Dim hr As Long
hr = Server.ConnectToServer ( )
If (hr = 0) Then
MsgBox ("Connected to IntegrationObjects.AE.Simulation Server")
End If	

2.2. Disconnect

Disconnect	
Disconnect from an AE Server that is already connected.	
LONG Disconnect (void) ; Public Function Disconnect ( ) As Long	
Return Value	**Description**
hr	The operation result.
Example	

Dim hr As Long
hr = Server.Disconnect ( )
If (hr = 0) Then
MsgBox ("Disconnected from IntegrationObjects.AE.Simulation Server")
End If	

2.3. CreateEventSubscription

CreateEventSubscription

Create an event subscription object.	
IDispatch* CreateEventSubscription(VARIANT_BOOL Active, LONG BufferTime, LONG* RevisedBufferTime); Public Function CreateEventSubscription(ByVal Active As Boolean, _ ByVal BufferTime As Long, _ RevisedBufferTime As Long) As Object	
Argument	**Description**
Active	Initial state of the event subscription, if true, the subscription will start sending events immediately.
BufferTime	Time in milliseconds that the server can hold back the notification to buffer multiple events (e.g. 1000 would indicate that the server can buffer events for one second and send the set of events as one network call back).
RevisedBufferTime	The revised buffer time that the server will actually support.
Return Value	**Description**
EventSub	The created event subscription object.
Example	

Dim ESub As New IOEventClientLib.IOEventSubscription
Dim RevisedBufTime As Long
Set ESub = MyServer.CreateEventSubscription(True, 1000,RevisedBufTime)
If Not (ESub Is Nothing) Then
…
End If	

2.4. QueryEventCategories

QueryEventCategories	
Retrieve the available event categories for a set of event types.	
LONG QueryEventCategories(IO_OPCAE_EVENTTYPES_CONSTANTS EventType); Public Function QueryEventCategories(ByVal EventType As Long) As Long	
Argument	**Description**

EventType	The event type taken from the **IO_OPCAE_EVENTTYPES_CONSTANTS** enumeration

Return Value	Description
CatgeoriesCount	The returned long value is the number of received event categories. If the method failed, it returns 0. Apart from the returned argument, the **CategoryID** and **CategoryDescription** Arrays will be filled respectively with category IDs and category descriptions.

Example

```
Dim count As Long
count = MyServer.QueryEventCategories(IO_OPCAE_ALL_EVENTS)
If (count > 0) Then
   Dim i As Integer
   ListView1.Clear
   For i = 0 To count - 1
      Dim lItem As ListItem
      Set lItem = ListView1.ListItems.Add(i + 1, , CStr
(MyServer.CategoryID( i)))
       lItem.SubItems(1) = MyServer.CategoryDescription (i)
   Next
End If

'The EventType could be also a set of event types as follows :

Dim EventType as Long
EventType = IO_OPCAE_CONDITION_EVENT And _
IO_OPCAE_SIMPLE_EVENT
count = MyServer.QueryEventCategories (EventType)
```

2.5. CategoryID

CategoryID	
Returned array of event category ID's.	
Public CategoryID (ByVal Index As Long) As Long	
Argument	**Description**
Index	The index of the requested category. The index should be an integer between 0 and the categories count returned by the

	QueryEventCategories function.
Return Value	**Description**
ID	The returned CategoryID given by the index.
Example	
MsgBox CStr (MyServer.**CategoryID** (2))	

2.6. CategoryDescription

CategoryDescription	
Returned array of event category Descriptions.	
Public CategoryDescription (ByVal Index As Long) As String	
Argument	**Description**
Index	The index of the requested category. The index should be an integer between 0 and the categories count returned by the **QueryEventCategories** function.
Return Value	**Description**
Description	The returned Category Description given by the index.
Example	
MsgBox CStr (MyServer.**CategoryDescription** (2))	

2.7. QueryEventAttributes

QueryEventAttributes	
Retrieves the list of available event attributes for a given event category.	
IDispatch* QueryEventAttributes (LONG EventCategory); Public Function QueryEventAttributes (ByVal EventCategory As Long) As Object	
Argument	**Description**
EventCategory	The given event category ID.
Return Value	**Description**

Object	The event attributes object : **IOEventAttributes** (details in IOEventAttributes Section)

Example

```
Dim eAttr As IOEventClientLib.IOEventAttributes
Set eAttr = MyServer.QueryEventAttributes (EventCategory)

 Dim j As Integer
 If (eAttr.AttributesCount <= 0) Then
      eAttributesForm.Caption = "No Attributes."
Else
      For j = 0 To eAttr.AttributesCount - 1
           Dim lItem As ListItem
           Set lItem = eAttributesForm.ListView1.ListItems.Add(j + 1, , _
           CStr (eAttr.AttributeIDs (j)))

           lItem.SubItems(1) = eAttr.AttributeDescriptions (j)
           lItem.SubItems(2) = eAttr.VarTypeString (j)
       Next
End If
```

2.8. AckCondition

AckCondition

Acknowledge the alarm condition.

LONG AckCondition (BSTR AcknowledgerID, BSTR Comment, BSTR Source, BSTR ConditionName, DATE ActiveTime, LONG ActiveTimeMilliseconds, LONG Cookie);

Public Function AckCondition (ByVal AcknowledgerID As String, ByVal Comment As String, ByVal Source As String, ByVal ConditionName As String, ByVal ActiveTime As Date, ByVal ActiveTimeMilliseconds As Long, ByVal Cookie As Long) As Long

Argument	Description
AcknowledgerID	Identifying the name of acknowledging party.
Source	The fully qualified source name.
Comment	The comment provided by the acknowledger.
ConditionName	Condition name.

ActiveTime	The time the condition went active.
ActiveTimeMilliseconds	The milliseconds of the active time.
Cookie	The cookie value of the received event.

Return Value	Description
hr	The operation result.

Example

```
…
 lngResult = MyServer.AckCondition(ID, Comment, Source, Condition,
ActiveTime, ActiveTimeMilliseconds, Cookie)
 If (lngResult = 0) Then
    MsgBox "Acknowledged"
End If
```

2.9. EnableConditionByArea

EnableConditionByArea

Enable condition alarm events in an area.

```
LONG EnableConditionByArea(BSTR Area);

Public Function EnableConditionByArea (ByVal Area As String) As Long
```

Argument	Description
Area	Enable alarm area.

Return Value	Description
hr	The operation result.

Example

```
Dim Area As String
Area = "…."
LngResult = MyServer.EnableConditionByArea ( Area)
```

2.10. EnableConditionBySource

EnableConditionBySource

Enable condition alarm events for a given source.

LONG EnableConditionBySource(BSTR Source); Public Function EnableConditionBySource (ByVal Source As String) As Long	
Argument	**Description**
Source	Enable alarm source.
Return Value	**Description**
hr	The operation result.

Example
Dim Source As String Source = “….” LngResult = MyServer.EnableConditionBySource (Source)

2.11. DisableConditionByArea

DisableConditionByArea	
Disable condition alarm events in an area.	
LONG DisableConditionByArea(BSTR Area); Public Function DisableConditionByArea (ByVal Area As String) As Long	
Argument	**Description**
Area	Disable alarm area.
Return Value	**Description**
hr	The operation result.
Example	
Dim Area As String Area = “….” LngResult = MyServer.DisableConditionByArea (Area)	

2.12. DisableConditionBySource

DisableConditionBySource	
Disable condition alarm events for a given source.	
LONG DisableConditionBySource(BSTR Source); Public Function DisableConditionBySource (ByVal Source As String) As Long	
Argument	**Description**
Source	Disable alarm source.
Return Value	**Description**

hr	The operation result.

Example

```
Dim Source As String
Source = "…."
LngResult = MyServer.DisableConditionBySource ( Source)
```

2.13. QueryConditionNames

QueryConditionNames

Retrieve the list of condition names for a specific event category.

```
LONG QueryConditionNames (LONG EventCategory);
Public Function QueryConditionNames (EventCategory As Long) As Long
```

Argument	Description
EventCategory	The event category ID.

Return Value	Description
Count	The number of the received condition names. You can read the returned array of condition names by the **ConditionName** function.

Example

```
count = MyServer.QueryConditionNames(EventCategory)
If (count > 0) Then
        Dim i As Integer
        LstConditions.Clear
        For i = 0 To count - 1
             lstConditions.AddItem MyServer.ConditionName (i)
         Next
 End If
```

2.14. ConditionName

ConditionName

A property defining the returned array of condition names.

```
BSTR ConditionName (LONG Index);
Public Property  ConditionName (ByVal Index As Long) As String
```

Argument	Description
Index	The index of the requested condition names. The index should be an integer between 0 and the conditions count returned by the **QueryConditionNames** function.
Return Value	**Description**
CondName	Returned Condition Name.
Example	
MsgBox MyServer.**ConditionName** (2)	

2.15. QuerySubConditionNames

QuerySubConditionNames	
Retrieves the list of subcondition names for a specific condition name.	
LONG QuerySubConditionNames (BSTR CondName); Public Function QuerySubConditionNames (CondName As String) As Long	
Argument	**Description**
CondName	The condition name.
Return Value	**Description**
Count	The number of the received subcondition names. You can read the returned array of subcondition names with the **SubConditionName** function.

Example

count = MyServer.**QuerySubConditionNames** (CondName)
If (count > 0) Then
Dim i As Integer
LstSubConditions.Clear
For i = 0 To count - 1
lstSubConditions.AddItem MyServer.**SubConditionName** (i)
Next
End If

2.16. SubConditionName

SubConditionName	
A property defining the returned array of subcondition names.	
BSTR SubConditionName (LONG Index); Public Property SubConditionName (ByVal Index As Long) As String	
Argument	**Description**
Index	The index of the requested subcondition name. The index should be an integer between 0 and the subcondition's count returned by the **QuerySubConditionNames** function.
Return Value	**Description**
SubCondName	Returned SubCondition Name.
Example	
MsgBox MyServer.**ConditionName** (''Between'')	

2.17. QuerySourceConditionNames

QuerySourceConditionNames	
Retrieve the list of condition names for a given source name.	
LONG QuerySourceConditionNames (BSTR Source); Public Function QuerySourceConditionNames (Source As String) As Long	
Argument	**Description**
Source	The source name.
Return Value	**Description**
Count	The number of the received condition names. You can read the returned array of condition names with the **SourceConditionName** function.
Example	

```
count = MyServer.QuerySourceConditionNames(EventCategory)
If (count > 0) Then
        Dim i As Integer
        LstConditions.Clear
        For i = 0 To count - 1
             lstConditions.AddItem MyServer.SourceConditionName (i)
         Next
 End If
```

2.18. SourceConditionName

SourceConditionName	
A property defining the returned array of source condition names.	
BSTR SourceConditionName (LONG Index); Public Property SourceConditionName (ByVal Index As Long) As String	
Argument	**Description**
Index	The index of the requested source condition name. The index should be an integer between 0 and the subcondition's count returned by the **QuerySourceConditionNames** function.
Return Value	**Description**

SourceCondName	Returned source condition name.
Example	
MsgBox MyServer.**SourceConditionName** (3)	

2.19. GetConditionState

GetConditionState	
Get condition state object for a given source's condition.	
IDispatch* GetConditionState(BSTR Source, BSTR ConditionName, LONG* AttributeIDs, LONG AttributeIDsCount); Public Function GetConditionState (ByVal Source As String, ByVal ConditionName As String, AttributeIDs As Long, ByVal AttributeIDsCount As Long) As IOEventConditionState	
Argument	**Description**
Source	Source name.
ConditionName	Condition name.
AttributeIDs	Array of requested attributes IDs.
AttributeIDsCount	Number of attributes IDs.
Return Value	**Description**
Object	An IOEventConditionState object.

2.20. CreateBrowser

CreateBrowser	
Create an event browser object for the server.	
IDispatch* CreateBrowser(void); Public Function CreateBrowser ( ) As IOEventBrowser	
Return Value	**Description**
Browser	This method returns the IOEventBrowser object related to the connected server.
Example	
Dim Browser As New IOEventBrowser Set Browser = MyServer..CreateBrowser	

3. IOEventSubscription

Using the following properties, the user can set up filtering options for event subscription and retrieve the events.

Property	Description
FilterEventType	(Read/Write) A Bit mask indicating the selected event types that should be sent to this subscription, built by combining values from the **IO_OPCAE_EVENTTYPES_CONSTANTS** enumeration. Only events satisfying the criterion "FilterEventType" will be returned.
FilterHighSeverity	(Read/Write) The Highest Severity limit for this subscription (Between 1 and 1000) and must be equal to or greater than FilterLowSeverity.
FilterLowSeverity	(Read/Write) The Low Severity limit for the subscription (Between 1 and 1000) and must be equal to or less than FilterHighSeverity.
FilterCategoriesCount	(Read-only) The number of category IDs. If 0, all categories are selected.

FilterSourcesCount	(Read-only) The number of sources. If 0, all sources are selected.
FilterAreasCount	(Read-only) The number of areas. If 0, all areas are selected.
FilterCategories	(Read-only) Array of Category ID's (see QueryEventCategories method) that should be sent to the event subscription object.
FilterAreas	(Read-only) List of area names.
FilterSources	(Read-only) List of source names.

3.1. GetSubscriptionState

GetSubscriptionState	
Get subscription state information.	
void GetSubscriptionState(VARIANT_BOOL* Active, LONG* BufferTime, LONG* MaxSize, LONG* hClientSubscription); Public Sub GetSubscriptionState (Active As Boolean, BufferTime As Long, MaxSize As Long, hClientSubscription As Long)	
Argument	**Description**
Active	When Active is true, the event subscription is active and it sends the OnEvent notifications.
BufferTime	Time in milliseconds that the server can hold back the notification to buffer multiple events.
MaxSize	The maximum number of events to buffer before sending events.
hClientSubscription	The client handles for the subscription.
Example	
Call MyEventSub.GetSubscriptionState (Active, BufferTime, MaxSize, hClientSubscription) MsgBox CStr (BufferTime)	

3.2. SetSubscriptionState

SetSubscriptionState
Modify the subscription state information.

void SetSubscriptionState(VARIANT_BOOL Active, LONG BufferTime, LONG MaxSize, LONG hClientSubscription, LONG* RevisedBufferTime, LONG* RevisedMaxSize);

Public Sub SetSubscriptionState (ByVal Active As Boolean, ByVal BufferTime As Long, ByVal MaxSize As Long, ByVal hClientSubscription As Long, RevisedBufferTime as Long, RevisedMaxSize As Long)

Argument	Description
Active	When Active is true, the event subscription is active and it sends the OnEvent notifications.
BufferTime	Time in milliseconds that the server can hold back the notification to buffer multiple events.
MaxSize	The maximum number of events to buffer before sending events.
hClientSubscription	The client handles for the subscription.
RevisedBufferTime	The revised buffer time that the server will support.
RevisedMaxSize	The revised maximum size that the server will support.

Example

Call MyEventSub.SetSubscriptionState (True, 1000, 3000, 1, RevisedBufferTime, RevisedMaxSize)

MsgBox CStr (RevisedBufferTime)

3.3. Refresh

Refresh

Refresh the subscription event.

void Refresh (void);

Public Sub Refresh ( )

3.4. CancelRefresh

CancelRefresh

Cancel the Refresh of the subscription event.

void CancelRefresh (void);

Public Sub CancelRefresh ( )

3.5. Activate

Activate
Activate the subscription event.
void Activate (void); Public Sub Activate ( )

3.6. Deactivate

Deactivate
Deactivate the subscription event.
void Deactivate (void); Public Sub Deactivate ( )

3.7. SelectAllAttributesForAllCatgories

SelectAllAttributesForAllCatgories
Selects all event attributes for all categories to be returned.
void SelectAllAttributesForAllCatgories (void); Public Sub SelectAllAttributesForAllCatgories ( )

3.8. GetReturnedEventAttributes

GetReturnedEventAttributes	
Selects all event attributes for all categories to be returned.	
IDispatch* GetReturnedEventAttributes (LONG EventCategory); Public Function GetReturnedEventAttributes (ByVal EventCategory As Long) As IOEventAttributes	
Argument	**Description**
EventCategory	The event Category ID for which the caller wishes to get the returned attributes.
Return Value	**Description**
IOEventAttributes Object	An IOEventAttributes object.

3.9. SelectReturnedEventAttributes

GetReturnedEventAttributes	
Selects event attributes for a specific category to be returned.	
void SelectReturnedEventAttributes (LONG EventCategory, LONG* AttributeIDs, LONG Count); Public Sub SelectReturnedEventAttributes (ByVal EventCategory As Long , AttributeIDs As Long, Count As Long)	
Argument	**Description**
EventCategory	The event Category ID.
AttributeIDs	The array of attribute ID's.
Count	The number of attributes.

3.10. GetFilter

GetFilter
Get OPC Filter options for this event subscription.
void GetFilter (void); Public Sub GetFilter ( )
Example

```
Call  MyEventSub.GetFilter

Text1.Text = Str (MyEventSub.FilterLowSeverity)
Text2.Text = Str (MyEventSub.FilterHighSeverity)

 List1.Clear
 List2.Clear
 Dim i As Integer
 If (MyEventSub.FilterSourcesCount > 0) Then
      For i = 0 To  MyEventSub.FilterSourcesCount - 1
         List2.AddItem  MyEventSub.FilterSources (i)
      Next i
 End If

If (MyEventSub.FilterAreasCount > 0) Then
      For i = 0 To MyEventSub.FilterAreasCount - 1
         List1.AddItem MyEventSub.FilterAreas (i)
      Next i
End If
```

3.11. ApplyFilter

ApplyFilter
Set up the filtering options for this event subscription.
void ApplyFilter (void); Public Sub ApplyFilter ( )
Example

MyEventSub.**FilterHighSeverity** = 1000
MyEventSub.**FilterLowSeverity** = 800
MyEventSub.**FilterEventType** = IO_OPCAE_CONDITION_EVENT
 	
MyEventSub.**ClearFilterAreas**
MyEventSub.**ClearFilterSources**
 	
Dim i As Integer
For i = 0 To List1.ListCount - 1
  MyEventSub.**AddFilterArea** (List1.List(i))
Next
 	
For i = 0 To List2.ListCount - 1
  MyEventSub.**AddFilterSource** (List2.List(i))
Next
 	
Call MyEventSub.**ApplyFilter**

3.12. AddFilterSource

AddFilterSource	
Add a source name to the array of filtering source names.	
void AddFilterSource (BSTR Source); Public Sub AddFilterSource (ByVal Source As String)	
Argument	**Description**
Source	The source name added to the filter source list.
Example	
Dim Source As String Source = “Tag.AE.1” AddFilterSource (Source)	

3.13. AddFilterArea

AddFilterArea	
Add an area name to the array of filtering area names.	
void AddFilterArea (BSTR Area); Public Sub AddFilterArea (ByVal Area As String)	
Argument	**Description**
Area	The area name added to the filter areas list.
Example	
Dim Area As String Area = “computer” AddFilterArea (Area)	

3.14. AddFilterCategory

AddFilterCategory	
Add a category ID to the array of filtering categories IDs.	
void AddFilterCategory (LONG CategoryID); **Public Sub AddFilterCategory (ByVal CategoryID As Long)**	
Argument	**Description**
CategoryID	The category ID
Example	
Dim ID As Long ID = 346 AddFilterCatgeory (ID)	

3.15. RemoveFilterSource

RemoveFilterSource	
Remove a source name from the source list.	
void RemoveFilterSource (LONG Index); Public Sub RemoveFilterSource (ByVal Index As Long)	
Argument	**Description**
Index	A given index that references the index of the source in the list to be removed. The index is between 0 and the source count –1.
Example	
RemoveFilterSource (1)	

3.16. RemoveFilterArea

RemoveFilterArea
Remove the area name from the area list.
void RemoveFilterArea (LONG Index); Public Sub RemoveFilterArea (ByVal Index As Long)

Argument	Description
Index	A given index that references the index of the area in the list to be removed. The index is between 0 and the area count –1.
Example	
RemoveFilterArea (1)	

3.17. RemoveFilterCategory

RemoveFilterCategory	
Remove the category ID from the category list.	
void RemoveFilterCategory (LONG Index); Public Sub RemoveFilterCategory (ByVal Index As Long)	
Argument	**Description**
Index	A given index that references the index of the category in the list to be removed. The index is between 0 and the category count –1.
Example	
RemoveFilterCatgeory (1)	

3.18. ClearFilterSources

ClearFilterSources
Clear the source list.
void ClearFilterSources (void); Public Sub ClearFilterSources ( )

3.19. ClearFilterAreas

ClearFilterAreas
Clear the area list.
void ClearFilterAreas (void); Public Sub ClearFilterAreas ( )

3.20. ClearFilterCategories

ClearFilterCategories
Clear the category list.
void ClearFilterCategories (void); Public Sub ClearFilterCategories ( )

4. IOEventConditionState

Property	Description
Acknowledged	(Read-only) True if the condition is acknowledged.
AcknowledgerID	(Read-only) The ID string for the last acknowledger of the condition.
Active	(Read-only) True if the condition is active.
ActiveSubCondition	(Read-only) The name of the active subcondition.

Comment	(Read-only) The comment string provided by the last acknowledger of the condition.
Quality	(Read-only) The OPC quality of the condition source.
LastAckTime	(Read-only) The time the condition was last acknowledged.
State	(Read-only) The condition state.
ActiveSubConditionDefinition	(Read-only) The active subcondition definition.
ActiveSubConditionDescription	(Read-only) The active subcondition description.
ActiveSubConditionSeverity	(Read-only) The active subcondition severity.
SubConditionLastActiveTime	(Read-only) The time that the subcondition last became active.
ConditionLastActiveTime	(Read-only) The time that the condition last became active.
ConditionLastInactiveTime	(Read-only) The time that the condition last became inactive.
StateString	(Read-only) A string representation of the condition state.
SubConditionsCount	(Read-only) Number of subconditions.
QualityString	(Read-only) A string representation of quality string.
SubConditionDefinitions	(Read-only) The definitions of the condition's subconditions.
SubConditionDescriptions	(Read-only) The descriptions of the condition's subconditions.
SubConditionNames	(Read-only) The names of the condition's subconditions.
SubConditionSeverities	(Read-only) The severities of the condition's subconditions.

5. IOEvent

Property	Description
AckRequired	(Read-only) A Boolean that defines whether the alarm condition requires Acknowledgment or not.

ActiveTime	(Read-only) The time the condition alarm went active.
EventTime	(Read-only) The time when the event occurred.
ConditionName	(Read-only) The condition name (condition events).
SubConditionName	(Read-only) The subcondition name (condition events).
EventCategory	(Read-only) Event category ID.
Cookie	
Message	(Read-only) The alarm message.
NewState	(Read-only) Bit mask that indicates the condition's current state, based on **IO_OPCAE_CONDITIONSTATE_CONSTANTS** (Enumerations section)
Quality	(Read-only) The quality value for condition events.
Severity	(Read-only) The severity value for condition events.
Source	(Read-only) The source item that caused the event to occur.
EventType	(Read-only) The event type. The allowable values are values from the enumeration **IO_OPCAE_EVENTTYPES_CONSTANTS**. (Enumerations section)
ActorID	(Read-only) The acknowledger name (condition events).
ChangeMask	(Read-only) Bit mask indicating which items have changed. This property is based on **IO_OPCAE_CHANGE_CONSTANTS** (Enumerations section)

StrQuality	(Read-only) String representation of condition event quality.
EventTimeMilliseconds	(Read-only) The milliseconds in the event time.
ActiveTimeMilliseconds	(Read-only) The milliseconds in the active time.
EventAttributesCount	(Read-only) The number of event attributes.
EventAttributes	(Read-only) Array of event attributes' values.
EventAttributesAsString	(Read-only) Array of event attributes' values as string.

6. IOEventServerStatus

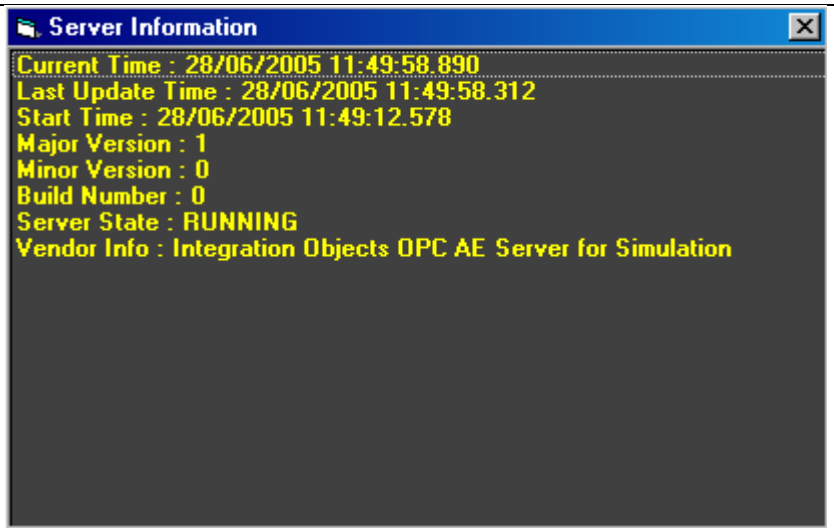
Property	Description
ServerState	(Read-only)The current status of the AE Server. Allowable values are given by the enumeration : IO_OPCAE_SERVERSTATE_CONSTANTS (see the following table).
CurrentTime	(Read-only)The current time of the OPC AE Server.
StartTime	(Read-only)The time when the server process started.
LastUpdateTime	(Read-only)The last updated time of the server process.
MajorVersion	(Read-only)The major version identification of the historian.
MinorVersion	(Read-only)The minor version identification of the historian.
BuildNumber	(Read-only)The build number identification of the historian
ServeStateString	(Read-only)A string explaining Server status instead of enumeration values.
VendorInfo	(Read-only)The vendor information for the OPC Server.
CurrentTimeMilliseconds	(Read-only)The milliseconds of the current time of the OPC AE Server.
StarttimeMilliseconds	(Read-only)The milliseconds of the time when the server process started.
LastUpdateTimeMilliseconds	(Read-only)The last updated time in milliseconds.

Example
Dim SStatus As New IOEventClientLib.IOEventServerStatus Set SStatus =.MyServer.ServerStatus If Not (SStatus Is Nothing) Then lstStatus.Clear Dim T as String T = CStr (SStatus.CurrentTime) + "." + CStr (SStatus.CurrentTimeMilliseconds) lstStatus.AddItem "Current Time : " + T

```
        T = CStr (SStatus.LastUpdateTime) + "." + CStr
(SStatus.LastUpdateTimeMilliseconds)
        lstStatus.AddItem "Last Update Time : " + T

        T = CStr (SStatus.StartTime) + "." + CStr (SStatus.StartTimeMilliseconds)
        lstStatus.AddItem "Start Time : " + T

        lstStatus.AddItem "Major Version : " + CStr (SStatus.MajorVersion)
        lstStatus.AddItem "Minor Version : " + CStr (SStatus.MinorVersion)
        lstStatus.AddItem "Build Number : " + CStr (SStatus.BuildNumber)
        lstStatus.AddItem "Server State : " + SStatus.ServerStatusString
        lstStatus.AddItem "Vendor Info : " + SStatus.VendorInfo
        Select Case (Sstatus.ServerState)
                Case IO_OPCAE_STATUS_RUNNING: MsgBox  “Running”
                Case IO_OPCAE_STATUS_FAILED:      MsgBox  “Failed’
        End Select
End If
```



7. IOEventBrowser

Property	Description
SelectedArea	(Read-only) The selected area in the browser.
SelectedSource	(Read-only) The selected source in the browser.

7.1. ShowBrowserDialog

ShowBrowserDialog
This method allows users to access a browser dialog to browse all address space and to select the source or area you want.
LONG ShowBrowserDialog (void) ; Public Function ShowBrowserDialog ( ) As Long
Example

LngResult = Browser.ShowBrowserDialog ( )
If (LngResult = 0) Then
MsgBox Browser.SelectedArea
MsgBox Browser.SelectedSource
End If

8. IOEventAttributes

Property	Description
AttributesCount	(Read-only) The number of attributes in the structure.
AttributeIDs	(Read-only) Array of the attributes' IDs.
AttributeDescriptions	(Read-only) Array of the attributes' Descriptions.
AttributeVarTypes	(Read-only) Array of the attributes' Data Types.
VarTypeString	(Read-only) Array of the attributes' Data Types in a string format.

9. ActiveX Defined Enumerations

9.1. IO_OPCAE_SERVERSTATE_CONSTANTS Enumeration

Values
IO_OPCAE_STATUS_RUNNING
IO_OPCAE_STATUS_FAILED
IO_OPCAE_STATUS_NOCONFIG
IO_OPCAE_STATUS_SUSPENDED
IO_OPCAE_STATUS_TEST

IO_OPCAE_STATUS_COMM_FAULT

9.2. IO_OPCAE_FILTER_CONSTANTS Enumeration

Values
IO_OPCAE_FILTER_BY_EVENT
IO_OPCAE_FILTER_BY_CATEGORY
IO_OPCAE_FILTER_BY_SEVERITY
IO_OPCAE_FILTER_BY_AREA
IO_OPCAE_FILTER_BY_SOURCE

9.3. IO_OPCAE_EVENTTYPES_CONSTANTS Enumeration

Values
IO_OPCAE_SIMPLE_EVENT
IO_OPCAE_TRACKING_EVENT
IO_OPCAE_CONDITION_EVENT
IO_OPCAE_ALL_EVENTS

9.4. IO_OPCAE_CONDITIONSTATE_CONSTANTS Enumeration

Values
IO_OPCAE_CONDITION_ENABLED
IO_OPCAE_CONDITION_ACTIVE
IO_OPCAE_CONDITION_ACKED

9.5. IO_OPCAE_CHANGE_CONSTANTS Enumeration

Values
IO_OPCAE_CHANGE_ACTIVE_STATE
IO_OPCAE_CHANGE_ACK_STATE
IO_OPCAE_CHANGE_ENABLE_STATE
IO_OPCAE_CHANGE_QUALITY
IO_OPCAE_CHANGE_SEVERITY
IO_OPCAE_CHANGE_SUBCONDITION
IO_OPCAE_CHANGE_MESSAGE
IO_OPCAE_CHANGE_ATTRIBUTE

USING OPC EVENT CLIENT .NET ACTIVEX

This control is a custom control which allows Visual Basic and other OLE Container applications to quickly and easily access data from any OPC Alarms & Events Server. The control's features enable the controlling application to establish a DCOM connection to an OPC A&E Server located on a local or on a remote machine. The application can subscribe to messages from the server and can be used to configure filters to receive only those of interest.
In this chapter, we will present the properties and methods of all objects in this Control.

10. IOEventClientDotNetCtrl

10.1. Logging

Integration Objects' Event Client .Net ActiveX produces a log file named "OPCEventClientDotNetActiveX.LOG" that records errors and debugging information. If difficulties occur with the application, the log file can be extremely valuable for troubleshooting. When operations are running normally, the control will log very little information.

This log file is generated by default in the bin folder under the installation folder. This path can be modified by the user using the "OPCEventClientDotNetActiveXConfig.ini" configuration file incorporated by the toolkit which includes several logging parameters. These parameters all have default settings and can be changed at start-up by editing the configuration file.

To change this file:

1. Open OPCEventClientDotNetActiveXConfig.ini in a text editor.
2. Edit any of the parameters listed in the following tables:

Log Setting	Description	Default Value
Buffer Size	The maximum number of messages to be stored in the runtime memory before launching writes action in the hard disk. It must be greater than 100.	200
Trace Level	The type of log messages to be logged. The value can be Control, Error, Warning, Inform, and Debug.	Error
AutoAppend	Set to true to continue writing log messages in the existed log file or to false to create a new file.	TRUE
Log File Path	The path where the log file will be generated. It's set to the bin folder by default and can be modified by the user to change it to a customized one.	The Bin folder path
Auto SaveTimeout	Time to wait to read all messages from the buffer	10
Maximum Files	Maximum number of files	5

Table 8: Configuration File Properties

10.2. GetLocalOPCEventServers

GetLocalOPCEventServers	
Retrieve the list of local AE servers.	
listsrv = GetLocalOPCEventServers	
Return Value	**Description**
listsrv	An array containing the progIDs (names) of registered local OPC AE Servers.

10.3. GetOPCEventServers

GetLocalOPCEventServers	
Retrieve the list of local AE Servers or AE Servers located on a remote machine.	
listsrv = GetOPCEventServers(host)	
Argument	**Description**
host	The machine where the OPC Server is registered. Note that the host may be the IP address of the machine.
Return Value	**Description**
listsrv	An array containing the progIDs (names) of registered local OPC AE Servers.

10.4. CreateServer

CreateServer	
Add a new Server object to the IOEventClientDotNetCtrl ActiveX control.	
NewServer = CreateServer()	
Return Value	**Description**
NewServer	This method returns a pointer to the newly created OPCAEServer object.

10.5. GetServer

Servers	
This method sends a reference to an OPCAEServer object, given the server index.	
Server = GetServer(index)	
Argument	**Description**
index	The server index.
Return Value	**Description**
GetServer	This method returns a pointer to the OPCAEServer object given by the server index. If the server with the given index is not found, this method returns NULL. This method also returns NULL if the Evaluation period has expired.

10.6. AboutBox

AboutBox
Display the About Box for the control. The About Box contains the product name and version number as well as other information about the software and Integration Objects.
AboutBox()

10.7. InitializeEventCallback

InitializeEventCallback
Initialize the Event Callback.
void InitializeEventCallback()

11. OPCAEServer

Property	Description
ServerAddress	[Read/Write]The machine where the OPC Server is registered. Note that the host may be the IP address of the machine.
ServerProgID	[Read/Write]The OPC Server name (ProgID).
ServerStatus	[ReadOnly]The current OPC AE Server Status Structure. This Structure is an **IOEventServerStatus** object.
ServerIndex	[ReadOnly]A Boolean that defines the filter by Event.
SubscriptionsCount	[ReadOnly] the number of subscriptions.
CategoriesNumber	[ReadOnly]Returns the number of categories defined within the A&E Server once the **QueryEventCategories** has been called.
ConditionsNumber	[ReadOnly]Returns the number of conditions related to the event category once **QueryConditionNames** has been called.
SubConditionsNumber	[ReadOnly]Returns the number of subconditions related to the condition name once **QuerySubConditionNames** has been called.
SourceConditionsNumber	[ReadOnly]Returns the number of conditions related to the concerned source name once **QuerySourceConditionNames** has been called.

11.1. ConnectToOPCServer

ConnectToOPCServer	
Connect to the AE Server defined by *ServerProgID* and *ServerAddress* properties.	
int ConnectToOPCServer ()	
Return Value	**Description**

hr	The operation result.

11.2. DisconnectfromServer

Disconnect	
Disconnect from an AE Server that is already connected.	
int DisconnectFromServer()	
Return Value	**Description**
hr	The operation result.

11.3. CreateEventSubscription

CreateEventSubscription	
Create an event subscription object.	
OPCEventSubscription CreateEventSubscription(string SubscriptionName, bool Active, int BufferTime, int MaxSize,out int RevisedBufferTime, out int RevisedMaxSize)	
Argument	**Description**
SubscriptionName	The subscription name
Active	Initial state of the event subscription, if true, the subscription will start sending events immediately.
BufferTime	Time in milliseconds that the server can hold back the notification to buffer multiple events (e.g. 1000 would indicate that the server can buffer events for one second and send the set of events as one network call back).
MaxSize	The requested maximum number of events that will be sent in a single callback. A value of 0 means that there is no limit to the number of events that will be sent in a single callback.
RevisedBufferTime	The revised buffer time that the server will actually support.
RevisedMaxSize	The revised max size that the server will actually support.

Return Value	Description
EventSub	The created event subscription object.

11.4. QueryEventCategories

QueryEventCategories	
Retrieve the available event categories for a set of event types.	
int QueryEventCategories(int iEventType)	
Argument	**Description**
EventType	The event type taken from the class object **OPCEventConstants** (OPCEventConstants section)
Return Value	**Description**
CatgeoriesCount	The returned long value is the number of received event categories. If the method failed, it returns 0. Apart from the returned argument, the **CategoryID** and **CategoryDescription** Arrays will be filled respectively with category IDs and category descriptions.

11.5. GetCategoryID

GetCategoryID	
Returned array of event category ID's.	
int GetCategoryID(int Index)	
Argument	**Description**
Index	The index of the requested category. The index should be an integer between 0 and the categories count returned by the **QueryEventCategories** function.
Return Value	**Description**
ID	The returned CategoryID given by the index.

11.6. GetCategoryDescription

GetCategoryDescription	
Returned array of event category Descriptions.	
string GetCategoryDescription(int Index)	
Argument	**Description**
Index	The index of the requested category. The index should be an integer between 0 and the categories count returned by the **QueryEventCategories** function.
Return Value	**Description**
Description	The returned Category Description given by the index.

11.7. QueryEventAttributes

QueryEventAttributes	
Retrieves the list of available event attributes for a given event category.	
OPCEventAttributes QueryEventAttributes(int EventCategory)	
Argument	**Description**
EventCategory	The given event category ID.
Return Value	**Description**
Object	The event attributes object : **OPCEventAttributes** (details in OPCEventAttributes Section)

11.8. AckCondition

AckCondition	
Acknowledge the alarm condition.	
int AckCondition(string AcknowledgerID, string Comment, string Source, string ConditionName, long ActiveTime, int Cookie)	
Argument	**Description**
AcknowledgerID	Identifying the name of acknowledging party.
Source	The fully qualified source name.
Comment	The comment provided by the acknowledger.

ConditionName	Condition name.
ActiveTime	The time the condition went active.
ActiveTimeMilliseconds	The milliseconds of the active time.
Cookie	The cookie value of the received event.

Return Value	Description
hr	The operation result.

11.9. AckConditionActiveFileTimeAsString

AckCondition	
Acknowledge the alarm condition using the active file time string format. This method is used in the VB6 environment development.	
int AckConditionActiveFileTimeAsString (string strAcknowledgerID, string strComment, string strSource, string strConditionName, string strActiveFileTime, int iCookie)	
Argument	**Description**
strAcknowledgerID	Identifying the name of acknowledging party.
strSource	The fully qualified source name.
strComment	The comment provided by the acknowledger.
strConditionName	Condition name.
strActiveFileTime	The file time the condition went active.
ActiveTimeMilliseconds	The milliseconds of the active time.
iCookie	The cookie value of the received event.

Return Value	Description
hr	The operation result.

1.1. EnableConditionByArea

EnableConditionByArea	
Enable condition alarm events in an area.	
int EnableConditionByArea(string Area)	
Argument	**Description**
Area	Enable alarm area.

Return Value	Description
hr	The operation result.

1.2. EnableConditionBySource

EnableConditionBySource	
Enable condition alarm events for a given source.	
int EnableConditionBySource(string Source)	
Argument	**Description**
Source	Enable alarm source.
Return Value	**Description**
hr	The operation result.

1.3. DisableConditionByArea

DisableConditionByArea	
Disable condition alarm events in an area.	
int DisableConditionByArea(string Area)	
Argument	**Description**
Area	Disable alarm area.
Return Value	**Description**
hr	The operation result.

1.4. DisableConditionBySource

DisableConditionBySource
Disable condition alarm events for a given source.
int DisableConditionBySource(string Source)

Argument	Description
Source	Disable alarm source.
Return Value	**Description**
hr	The operation result.

1.5. QueryConditionNames

QueryConditionNames	
Retrieve the list of condition names for a specific event category.	
int QueryConditionNames(int EventCategory)	
Argument	**Description**
EventCategory	The event category ID.
Return Value	**Description**
Count	The number of the received condition names. You can read the returned array of condition names by the **ConditionName** function.

1.6. GetConditionName

GetConditionName	
A property defining the returned array of condition names.	
string GetConditionName(int Index)	
Argument	**Description**
Index	The index of the requested condition names. The index should be an integer between 0 and the conditions count returned by the **QueryConditionNames** function.
Return Value	**Description**
CondName	Returned Condition Name.

1.7. QuerySubConditionNames

QuerySubConditionNames	
Retrieves the list of subcondition names for a specific condition name.	
int QuerySubConditionNames(string strConditionName)	
Argument	**Description**
CondName	The condition name.
Return Value	**Description**
Count	The number of the received subcondition names. You can read the returned array of subcondition names with the **SubConditionName** function.

1.8. GetSubConditionName

GetSubConditionName	
A property defining the returned array of subcondition names.	
string GetSubConditionName(int Index)	
Argument	**Description**
Index	The index of the requested subcondition name. The index should be an integer between 0 and the subcondition's count returned by the **QuerySubConditionNames** function.
Return Value	**Description**
SubCondName	Returned SubCondition Name.

1.9. QuerySourceConditionNames

QuerySourceConditionNames	
Retrieve the list of condition names for a given source name.	
int QuerySubConditionNames(string strConditionName)	
Argument	**Description**
Source	The source name.
Return Value	**Description**
Count	The number of the received condition names. You can read the returned array of condition names with the **SourceConditionName** function.

1.10. GetSourceConditionName

GetSourceConditionName
A property defining the returned array of source condition names.
string GetSourceConditionName(int Index)

Argument	Description
Index	The index of the requested source condition name. The index should be an integer between 0 and the subcondition's count returned by the **QuerySourceConditionNames** function.

Return Value	Description
SourceCondName	Returned source condition name.

1.11. GetConditionState

GetConditionState	
Get condition state object for a given source's condition.	
OPCConditionState GetConditionState(string Source, string ConditionName, int EventCategory)	
Argument	**Description**
Source	Source name.
ConditionName	Condition name.
EventCategory	Event Category

1.12. GetCanFilterByEvent

GetCanFilterByEvent	
Check if the server can filter by event.	
bool GetCanFilterByEvent()	
Return Value	**Description**
boolean	This method returns the true if the server can filter by event and false if not

1.13. GetCanFilterBySeverity

GetCanFilterBySeverity
Check if the server can filter by severity.

bool GetCanFilterBySeverity()	
Return Value	**Description**
boolean	This method returns the true if the server can filter by severity and false if not

1.14. GetCanFilterByCategory

GetCanFilterByCategory	
Check if the server can filter by category.	
bool GetCanFilterByCategory()	
Return Value	**Description**
boolean	This method returns the true if the server can filter by category and false if not

1.15. GetCanFilterByArea

GetCanFilterByArea	
Check if the server can filter by area.	
bool GetCanFilterByArea()	
Return Value	**Description**
boolean	This method returns the true if the server can filter by area and false if not

1.16. GetCanFilterBySource

GetCanFilterBySource	
Check if the server can filter by source.	
bool GetCanFilterBySource()	
Return Value	**Description**
boolean	This method returns the true if the server can filter by source and false if not

1.17. CreateBrowser

CreateBrowser	
Create an event browser object for the server.	
OPCEventBrowser CreateBrowser()	
Return Value	**Description**
Browser	This method returns the OPCEventBrowser object related to the connected server.

1.18. GetCategoryDescriptionfromID

GetGategoryDescriptionfromID	
Return the category description for a given category ID once the **QueryEventCategories** has been called	
string GetCategoryDescriptionfromID(int ID)	
Return Value	**Description**
catDsc	This method returns the event category description related to a given category ID.

1.19. GetCategoryDescriptionfromID

GetGategoryDescriptionfromID	
Return the category description for a given category ID once the **QueryEventCategories** has been called	
string GetCategoryDescriptionfromID(int ID)	
Return Value	**Description**
catDsc	This method returns the event category description related to a given category ID.

1.20. GetCategoryIDfromDescription

GetCategoryIDfromDescription
Return the category ID for a given category description once the **QueryEventCategories** has been called

int GetCategoryIDfromDescription(string CatDesc)	
Return Value	**Description**
ID	This method returns the event ID related to a given category description.

2. OPCEventSubscription

Using the following properties, the user can set up filtering options for event subscription and retrieve the events.

Property	Description
SubscriptionName	(Read Only) The subscription name
FilterEventType	(Read/Write) A Bit mask indicating the selected event types that should be sent to this subscription, built by combining values from the class object **OPCEventConstants** (OPCEventConstants section). Only events satisfying the criterion “FilterEventType” will be returned.
FilterHighSeverity	(Read/Write) The Highest Severity limit for this subscription (Between 1 and 1000) and must be equal to or greater than FilterLowSeverity.
FilterLowSeverity	(Read/Write) The Low Severity limit for the subscription (Between 1 and 1000) and must be equal to or less than FilterHighSeverity.
FilterCategoriesCount	(Read-only) The number of category IDs. If 0, all categories are selected.
FilterSourcesCount	(Read-only) The number of sources. If 0, all sources are selected.
FilterAreasCount	(Read-only) The number of areas. If 0, all areas are selected.
FilterCategories	(Read-only) Array of Category ID's (see QueryEventCategories method) that should be sent to the event subscription object.
FilterAreas	(Read-only) List of area names.
FilterSources	(Read-only) List of source names.

2.1. GetSubscriptionState

GetSubscriptionState	
Get subscription state information.	
void GetSubscriptionState(out bool Active, out int BufferTime, out int MaxSize, out int hClientSubscription)	
Argument	**Description**
Active	When Active is true, the event subscription is active and it sends the OnIncomingEvent notifications.
BufferTime	Time in milliseconds that the server can hold back the notification to buffer multiple events.
MaxSize	The maximum number of events to buffer before sending events.
hClientSubscription	The client handles for the subscription.

2.2. SetSubscriptionState

SetSubscriptionState	
Modify the subscription state information.	
void SetSubscriptionState(bool Active, int BufferTime, int MaxSize, int hClientSubscription, out int RevisedBufferTime, out int RevisedMaxSize)	
Argument	**Description**
Active	When Active is true, the event subscription is active and it sends the OnEvent notifications.
BufferTime	Time in milliseconds that the server can hold back the notification to buffer multiple events.
MaxSize	The maximum number of events to buffer before sending events.
hClientSubscription	The client handles for the subscription.
RevisedBufferTime	The revised buffer time that the server will support.
RevisedMaxSize	The revised maximum size that the server will support.

2.3. Refresh

Refresh
Refresh the subscription event.
void Refresh()

2.4. CancelRefresh

CancelRefresh
Cancel the Refresh of the subscription event.
void CancelRefresh()

2.5. Activate

Activate
Activate the subscription event.
void Activate()

2.6. Deactivate

Deactivate
Deactivate the subscription event.
void Deactivate()

2.7. SelectAllAttributesForAllCatgories

SelectAllAttributesForAllCatgories
Selects all event attributes for all categories to be returned.
void SelectAllAttributesForAllCatgories()

2.8. GetReturnedEventAttributes

GetReturnedEventAttributes	
Selects all event attributes for all categories to be returned.	
OPCEventAttributes GetReturnedEventAttributes(int EventCategory)	
Argument	**Description**
EventCategory	The event Category ID for which the caller wishes to get the returned attributes.
Return Value	**Description**

OPCEventAttributes Object	An OPCEventAttributes object.

2.9. SelectReturnedEventAttributes

GetReturnedEventAttributes	
Selects event attributes for a specific category to be returned.	
void SelectReturnedEventAttributes(int EventCategory, int[] AttributeIDs, int Count)	
Argument	**Description**
EventCategory	The event Category ID.
AttributeIDs	The array of attribute ID's.
Count	The number of attributes.

2.10. GetFilter

GetFilter
Get OPC Filter options for this event subscription.
void GetFilter()

2.11. ApplyFilter

ApplyFilter
Set up the filtering options for this event subscription.
void ApplyFilter()

2.12. AddFilterSource

AddFilterSource	
Add a source name to the array of filtering source names.	
void AddFilterSource(string Source)	
Argument	**Description**

Source	The source name added to the filter source list.

2.13. AddFilterArea

AddFilterArea	
Add an area name to the array of filtering area names.	
void AddFilterArea(string Area)	
Argument	**Description**
Area	The area name added to the filter areas list.

2.14. AddFilterCategory

AddFilterCategory	
Add a category ID to the array of filtering categories IDs.	
void AddFilterCategory(int CategoryID)	
Argument	**Description**
CategoryID	The category ID

2.15. RemoveFilterSource

RemoveFilterSource	
Remove a source name from the source list.	
void RemoveFilterSource(int iIndex)	
Argument	**Description**
Index	A given index that references the index of the source in the list to be removed. The index is between 0 and the source count –1.

2.16. RemoveFilterArea

RemoveFilterArea
Remove the area name from the area list.
void RemoveFilterArea(int Index)

Argument	Description
Index	A given index that references the index of the area in the list to be removed. The index is between 0 and the area count –1.
Example	
RemoveFilterArea (1)	

2.17. RemoveFilterCategory

RemoveFilterCategory	
Remove the category ID from the category list.	
void RemoveFilterCategory(int Index)	
Argument	**Description**
Index	A given index that references the index of the category in the list to be removed. The index is between 0 and the category count –1.

2.18. ClearFilterSources

ClearFilterSources
Clear the source list.
void ClearFilterSources()

2.19. ClearFilterAreas

ClearFilterAreas
Clear the area list.
void ClearFilterAreas()

2.20. ClearFilterCategories

ClearFilterCategories
Clear the category list.
void ClearFilterCategories()

3. OPCConditionState

Property	Description
Acknowledged	(Read-only) True if the condition is acknowledged.
AcknowledgerID	(Read-only) The ID string for the last acknowledger of the condition.
Active	(Read-only) True if the condition is active.
ActiveSubCondition	(Read-only) The name of the active subcondition.

Comment	(Read-only) The comment string provided by the last acknowledger of the condition.
Quality	(Read-only) The OPC quality of the condition source.
LastAckTime	(Read-only) The time the condition was last acknowledged.
State	(Read-only) The condition state.
ActiveSubConditionDefinition	(Read-only) The active subcondition definition.
ActiveSubConditionDescription	(Read-only) The active subcondition description.
ActiveSubConditionSeverity	(Read-only) The active subcondition severity.
SubConditionLastActiveTime	(Read-only) The time that the subcondition last became active.
ConditionLastActiveTime	(Read-only) The time that the condition last became active.
ConditionLastInactiveTime	(Read-only) The time that the condition last became inactive.
State	(Read-only) The condition state.
SubConditionsCount	(Read-only) Number of subconditions.
QualityString	(Read-only) A string representation of quality string.
SubConditionDefinitions	(Read-only) The definitions of the condition's subconditions.
SubConditionDescriptions	(Read-only) The descriptions of the condition's subconditions.
SubConditionNames	(Read-only) The names of the condition's subconditions.
SubConditionSeverities	(Read-only) The severities of the condition's subconditions.

3.1. GetQuality

GetQuality
Return the quality value.
int GetQuality()

3.2. GetState

GetState
Return the state value.
int GetState()

3.3. GetActiveSubConditionDefinition

GetActiveSubConditionDefinition
Return the active subcondition definition string.
string GetActiveSubConditionDefinition()

3.4. GetActiveSubConditionDescription

GetActiveSubConditionDescription
Return the active subcondition description string
string GetActiveSubConditionDescription()

3.5. GetActiveSubConditionSeverity

GetActiveSubConditionSeverity
Return the active subcondition severity value.
string GetActiveSubConditionSeverity()

3.6. GetSubConditionLastActiveTime

GetSubConditionLastActiveTime
Return the subcondition last active time datetime.
DateTime GetSubConditionLastActiveTime()

3.7. GetConditionLastActiveTime

GetConditionLastActiveTime
Return the condition last active time datetime
DateTime GetConditionLastActiveTime( )

3.8. GetConditionLastInactiveTime

GetConditionLastInactiveTime
Return the condition last inactive time datetime
DateTime GetConditionLastInactiveTime ( )

3.9. GetStateString

GetStateString
Return the state string.
string GetStateString()

3.10. GetSubConditionsCount

GetSubConditionsCount
Return the subConditions count.
int GetSubConditionsCount()

3.11. GetSubConditionDefinitions

GetSubConditionDefinitions	
Return the subCondition definition string for a given index.	
string GetSubConditionDefinitions(int Index)	
Argument	**Description**
Index	A given index that references the index of the sub condition definition in the list to be returned. The index is between 0 and the sub condition definitions count –1.

3.12. GetSubConditionDescriptions

GetSubConditionDescriptions	
Return the subCondition description string for a given index.	
string GetSubConditionDescriptions(int Index)	
Argument	**Description**
Index	A given index that references the index of the sub condition descriptions in the list to be returned. The index is between 0 and the sub condition descriptions count –1.

3.13. GetSubConditionSeverities

GetSubConditionSeverities	
Return the subCondition severity value for a given index.	
int GetSubConditionSeverities(int Index)	
Argument	**Description**
Index	A given index that references the index of the sub condition severity in the list to be returned. The index is between 0 and the sub condition severities count –1.

3.14. GetSubConditionNames

GetSubConditionNames	
Return the subCondition name string for a given index.	
string GetSubConditionNames(int Index)	
Argument	**Description**
Index	A given index that references the index of the sub condition name in the list to be returned. The index is between 0 and the sub condition names count –1.

3.15. GetQualityString

GetQualityString

Return the quality string.
string GetQualityString()

3.16. GetSubConditionLastActiveTimeAsString

GetSubConditionLastActiveTimeAsString
Return the sub condition last active time string.
string GetSubConditionLastActiveTimeAsString()

3.17. GetConditionLastActiveTimeAsString

GetConditionLastActiveTimeAsString
Return the condition last active time string.
string GetConditionLastActiveTimeAsString()

3.18. GetSubConditionLastInactiveTimeAsString

GetSubConditionLastInactiveTimeAsString
Return the sub condition last inactive time string.
string GetConditionLastInactiveTimeAsString ()

3.19. GetLastAckTimeAsString

GetLastAckTimeAsString
Return the last acknowledgment time string.
string GetLastAckTimeAsString()

4. EventStruct

Property	Description
AckRequired	(Read-only) A Boolean that defines whether the alarm condition requires Acknowledgment or not.
ActiveTime	(Read-only) The time the condition alarm went active.
EventTime	(Read-only) The time when the event occurred.
ConditionName	(Read-only) The condition name (condition events).
SubConditionName	(Read-only) The subcondition name (condition events).
EventCategory	(Read-only) Event category ID.
Cookie	
Message	(Read-only) The alarm message.
NewState	(Read-only) Bit mask that indicates the condition's current state, based on the class object **OPCEventConstants** (OPCEventConstants section)
Quality	(Read-only) The quality value for condition events.
Severity	(Read-only) The severity value for condition events.
Source	(Read-only) The source item that caused the event to occur.
EventType	(Read-only) The event type. The allowable values are values from the class object **OPCEventConstants** (OPCEventConstants section)
ActorID	(Read-only) The acknowledger name (condition events).
ChangeMask	(Read-only) Bit mask indicating which items have changed. This property is based on the class object **OPCEventConstants** (OPCEventConstants section)
EventTimeMilliseconds	(Read-only) The milliseconds in the event time.
ActiveTimeMilliseconds	(Read-only) The milliseconds in the active time.
EventAttributesCount	(Read-only) The number of event attributes.
EventAttributes	(Read-only) Array of event attributes' values.
EventAttributesAsString	(Read-only) Array of event attributes' values as string.

4.1. GetEventAttributes

GetEventAttributes
Return the sub condition last active time string.

object GetEventAttributes(int Index)	
Argument	**Description**
Index	A given index that references the index of the event attribute object in the list to be returned. The index is between 0 and the event attributes count –1.

4.2. GetEventAttributesAsString

GetEventAttributesAsString	
Return the sub condition last active time string.	
string GetEventAttributesAsString(int Index)	
Argument	**Description**
Index	A given index that references the index of the event attribute string in the list to be returned. The index is between 0 and the event attributes string count –1.

4.3. GetStrQuality

GetStrQuality
Return the quality string.
string GetStrQuality()

4.4. GetEventTimeAsString

GetEventTimeAsString
Return the event time string.
string GetEventTimeAsString()

4.5. GetActiveTimeAsString

GetActiveTimeAsString
Return the active time string.

string GetActiveTimeAsString()

5. OPCServerStatus

Property	Description
ServerState	(Read-only)The current status of the AE Server. Allowable values are given by the enumeration : IO_OPCAE_SERVERSTATE_CONSTANTS (see the following table).
CurrentTime	(Read-only)The current time of the OPC AE Server.
StartTime	(Read-only)The time when the server process started.
LastUpdateTime	(Read-only)The last updated time of the server process.
MajorVersion	(Read-only)The major version identification of the historian.
MinorVersion	(Read-only)The minor version identification of the historian.
BuildNumber	(Read-only)The build number identification of the historian
VendorInfo	(Read-only)The vendor information for the OPC Server.
CurrentTimeMilliseconds	(Read-only)The milliseconds of the current time of the OPC AE Server.
StarttimeMilliseconds	(Read-only)The milliseconds of the time when the server process started.
LastUpdateTimeMilliseconds	(Read-only)The last updated time in milliseconds.

4.1. GetServerStatusString

GetServerStatusString
Return the server status string.
string GetServerStatusString()

4.2. GetStartTimeString

GetStartTimeString
Return the start time string.
string GetStartTimeString()

4.3. GetLastUpdateString

GetLastUpdateString
Return the last update time string.
string GetLastUpdateString()

4.4. GetCurrentTimeString

GetCurrentTimeString
Return the current time string.
string GetCurrentTimeString()

6. OPCEventBrowser

Property	Description
SelectedArea	(Read-only) The selected area in the browser.
SelectedSource	(Read-only) The selected source in the browser.

6.1. ShowBrowserDialog

ShowBrowserDialog
This method allows users to access a browser dialog to browse all address space and to select the source or area you want.
bool ShowBrowserDialog()

7. OPCEventAttributes

Property	Description
AttributesCount	(Read-only) The number of attributes in the structure.
AttributeIDs	(Read-only) Array of the attributes' IDs.
AttributeDescriptions	(Read-only) Array of the attributes' Descriptions.
AttributeVarTypes	(Read-only) Array of the attributes' Data Types.

7.1. GetAttributeIDs

GetAttributeIDs	
Return the attribute ID.	
int GetAttributeIDs (int Index)	
Argument	**Description**
Index	A given index that references the index of the event attribute ID in the list to be returned. The index is between 0 and the event attributes count –1.

7.2. GetAttributeDescriptions

GetAttributeDescriptions	
Return the attribute description.	
string GetAttributeDescriptions (int Index)	
Argument	**Description**
Index	A given index that references the index of the event attribute description in the list to be returned. The index is between 0 and the event attributes descriptions count –1.

7.3. GetAttributeVarTypes

GetAttributeVarTypes
Return the attribute vartype

int GetAttributeVarTypes (int Index)	
Argument	**Description**
Index	A given index that references the index of the event attribute vartypes in the list to be returned. The index is between 0 and the event attributes vartypes count –1.

7.4. GetVarTypeString

GetVarTypeString	
Return the attribute vartype string	
string GetVarTypeString (int Index)	
Argument	**Description**
Index	A given index that references the index of the event attribute vartype string in the list to be returned. The index is between 0 and the event attributes vartype string count –1.

8. OPCEventFilter

Property	Description
EventType	(Read\Write) the event type
LowSeverity	(Read\Write) the low severity
HighSeverity	(Read\Write) the high severity
FilterCategoriesCount	(Read\Write) the filter categories count
FilterAreasCount	(Read\Write) the filter areas count
FilterSourcesCount	(Read\Write) the filter sources count

8.1. SetEventCategories

SetEventCategories	
Set the event category of a given index	
void SetEventCategories(int Index, int NewVal)	
Argument	**Description**
Index	A given index that references the index of the event category ID in the list to be modified. The index is between 0 and the event

	categories count –1.
NewVal	The new value

8.2. SetFilterAreas

SetFilterAreas	
Set the area of a given index	
void SetFilterAreas(int Index, string NewVal)	
Argument	**Description**
Index	A given index that references the index of the area in the list to be modified. The index is between 0 and the event areas count –1.
NewVal	The new value

8.3. GetFilterAreas

GetFilterAreas	
Return the area of a given index	
string GetFilterAreas(int Index)	
Argument	**Description**
Index	A given index that references the index of the event area in the list to be returned. The index is between 0 and the event areas count –1.

8.4. GetEventCategories

GetEventCategories	
Return the event category of a given index	
int GetEventCategories(int Index)	
Argument	**Description**
Index	A given index that references the index of the event categories ID in the list to be returned. The index is between 0 and the event categories ID count –1.

8.5. GetFilterSources

GetFilterSources	
Return the source of a given index	
string GetFilterSources(int Index)	
Argument	**Description**
Index	A given index that references the index of the event source in the list to be returned. The index is between 0 and the event areas count –1.

8.6. SetFilterSources

SetFilterSources	
Set the source of a given index	
void SetFilterSources(int Index, string NewVal)	
Argument	**Description**
Index	A given index that references the index of the source in the list to be modified. The index is between 0 and the event sources count –1.
NewVal	The new value

9. OPCEventConstants

9.1. OPC FILTER CONSTANTS

Values
OPC_FILTER_BY_EVENT
OPC_FILTER_BY_CATEGORY
OPC_FILTER_BY_SEVERITY
OPC_FILTER_BY_AREA
OPC_FILTER_BY_SOURCE

9.2. OPC EVENT TYPES CONSTANTS

Values

OPC_SIMPLE_EVENT
OPC_TRACKING_EVENT
OPC_CONDITION_EVENT
OPC_ALL_EVENTS

9.3. OPC CONDITION STATE CONSTANTS

Values
OPC_CONDITION_ENABLED
OPC_CONDITION_ACTIVE
OPC_CONDITION_ACKED

9.4. OPC CHANGE CONSTANTS

Values
OPC_CHANGE_ACTIVE_STATE
OPC_CHANGE_ACK_STATE
OPC_CHANGE_ENABLE_STATE
OPC_CHANGE_QUALITY
OPC_CHANGE_SEVERITY
OPC_CHANGE_SUBCONDITION
OPC_CHANGE_MESSAGE
OPC_CHANGE_ATTRIBUTE

5. Deploying the Event Client .Net ActiveX in Microsoft Visual Basic 6.0

5.1. Create a Standard EXE

Run the Microsoft Visual Basic 6.0 and select a Standard EXE to create a new project as shown below.

Figure 114: Create a VB6 Standard EXE

5.2. Add the OPC Event Client .Net reference

1. Select the *Project* menu item and click on *References*

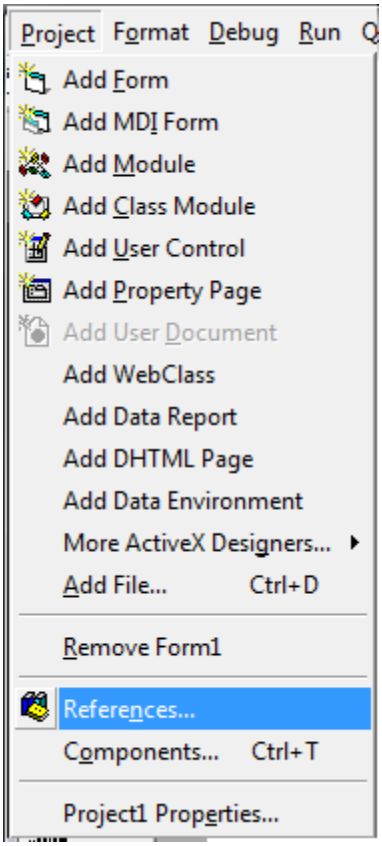


Figure 115: Select Projet Reference

2. Click on the Browse button and select the OPC Event Client tlb file path (.:\Program Files (x86)\Integration Objects\Integration Objects' OPC AE ActiveX\Bin\DotNet\ OPCEventClientActiveX)

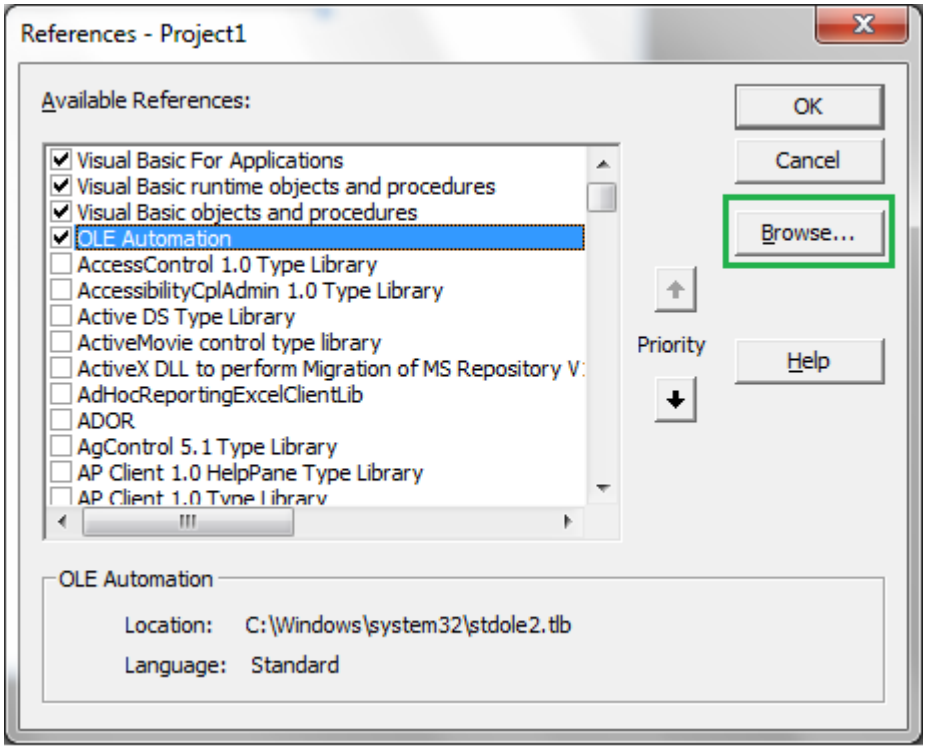


Figure 116: Browse the OPC AE Net Logger Path

3. Select the “IntegrationObjects.OPCEventClientDotNetActiveX.tlb” and click on the *Open* button.

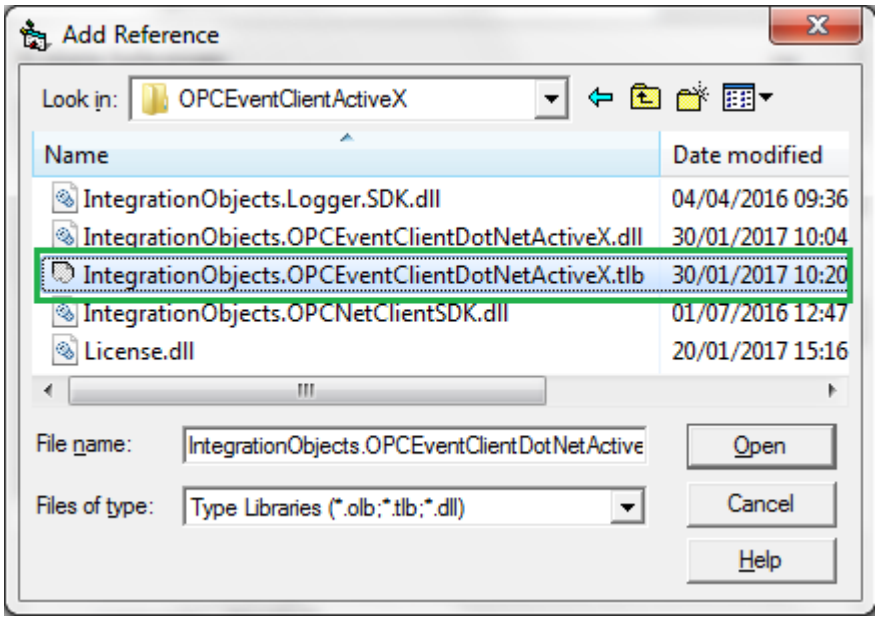


Figure 117: Select the type library (.tlb) file

4. Once the IntegrationObject_OPCEventClientDotNetActiveX reference is checked, click on OK

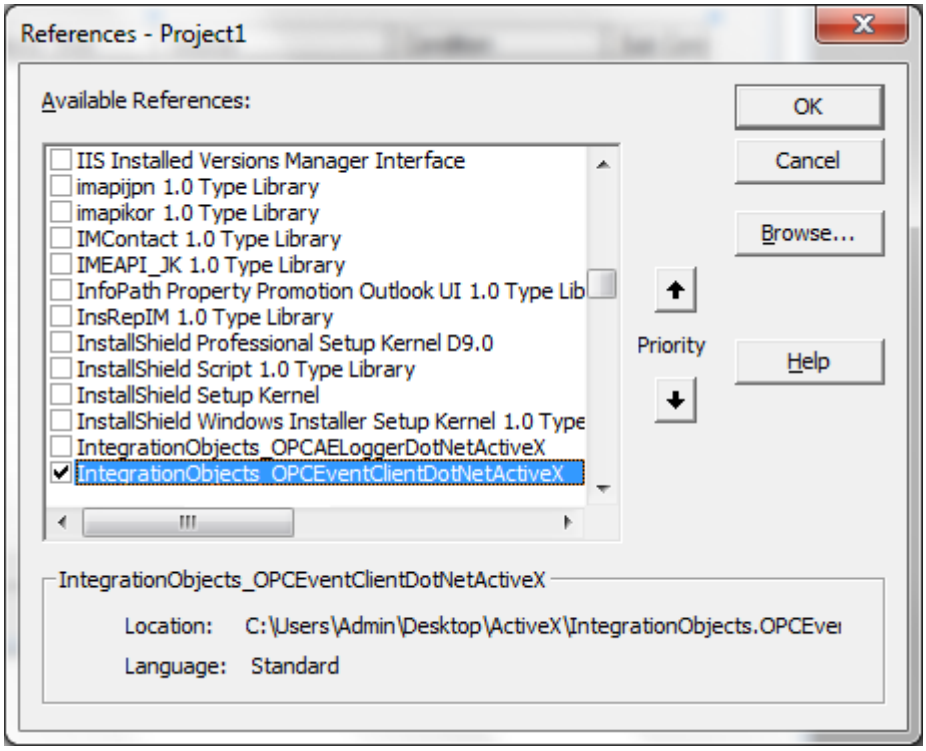


Figure 118: Check the OPC Event Client Net ActiveX Reference

5.3. Add the OPC Event Client Net ActiveX Component to the Toolbar

5. Select the *Project* menu item and click on *Components*

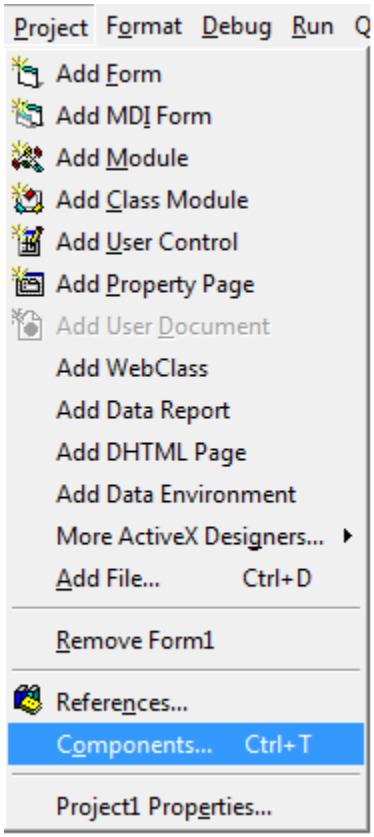


Figure 119: Select Projet Components

6. Check the IntegrationObject_OPCEventClientDotNetActiveX component then click on the *OK* button

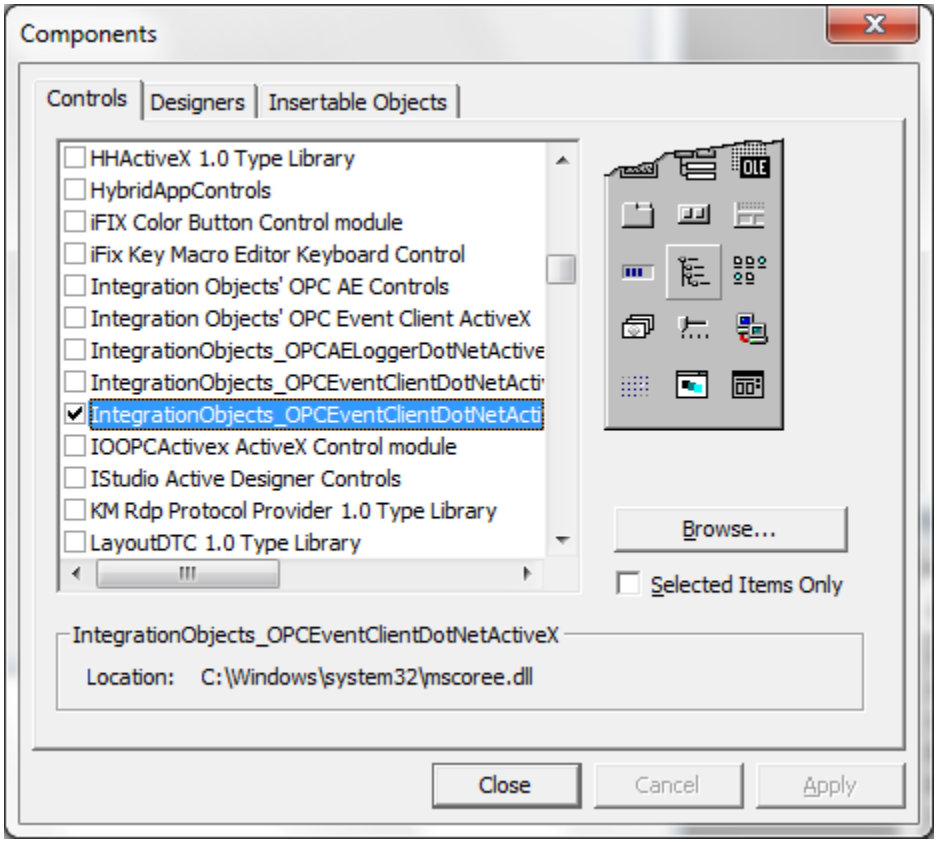


Figure 120: Select the OPC Event Net ActiveX Component

7. Select the IOEventClientDotNetCtrl already added in the toolbar and draw the control in the form as shown below

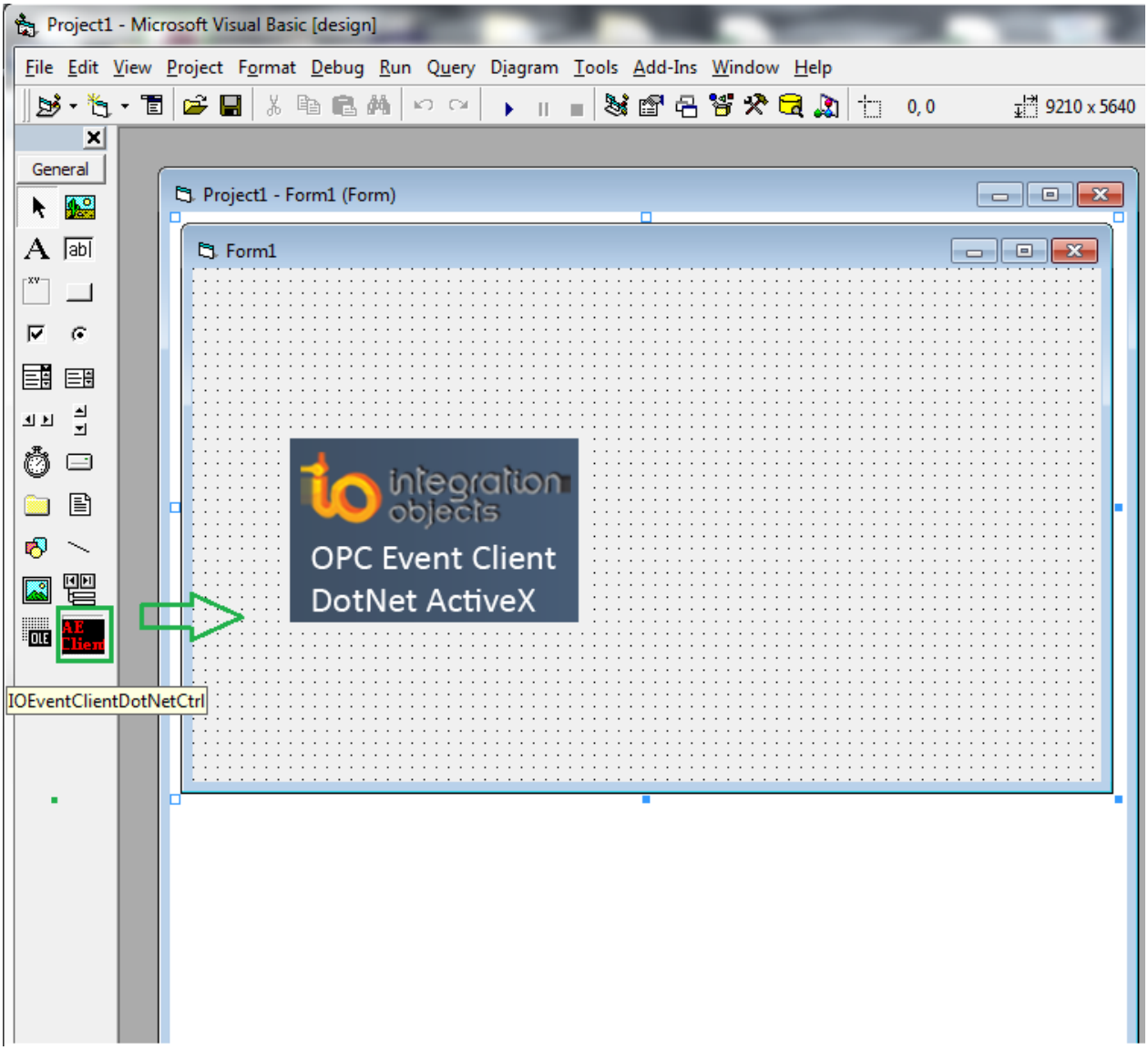


Figure 121: Add the OPC Event Client Net ActiveX Component to the Form

CONFIGURING DCOM

In order to retrieve data from OPC Server(s) in real time, Integration Objects' OPC AE ActiveX can be used in different configurations, including local and distributed configurations. In a local configuration, the toolkit and OPC Server(s) all run on the same computer. In that case, the installation process does not need any specific settings. In a distributed configuration, these components are executed on two or more computers cooperatively: Integration Objects' OPC AE ActiveX software initially resides on a remote computer (Client Computer) and uses the DCOM mechanism to directly access server(s). To enable this functionality, some additional settings are needed on both the remote server and the local client computer. This section is intended to provide general guidance on proper DCOM Config Utility settings for computers on which the OPC AE Controls and OPC server(s) are running.

1. Client Side DCOM Configuration

Step 1: Once you log in as an Administrator, setup Client machine with these instructions:

1. Choose the Run Option from the Windows' Start menu and type DCOMCNFG then click *OK* to run it. When you first launch the utility, it will look like this:

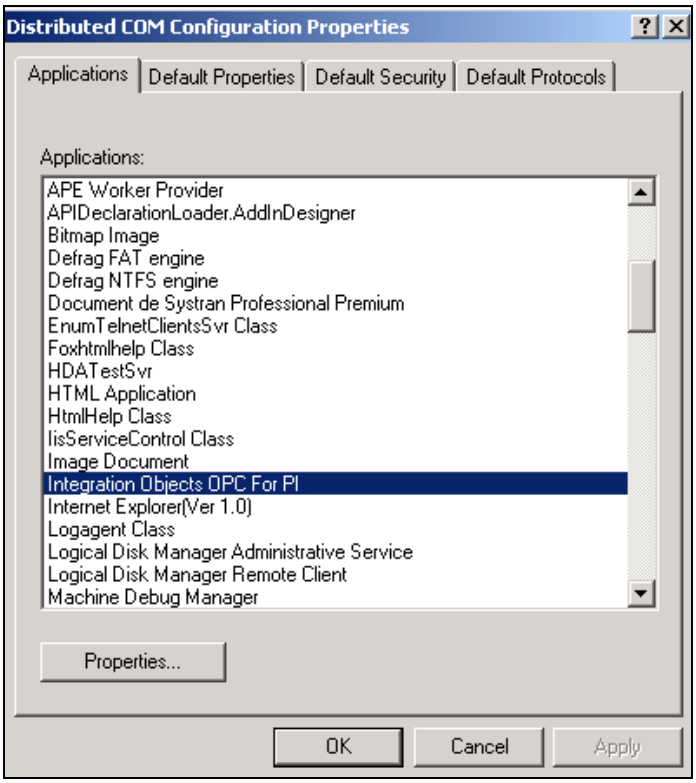


Figure 122:Distributed COM Configuration

2. DCOM Configuration Properties-Default Properties Tab:
    a. The **Enable Distributed COM** on this computer MUST be checked.
    b. The Default Authentication Level should be set to Connect.
    c. The Default Impersonation Level should be set to Identity.

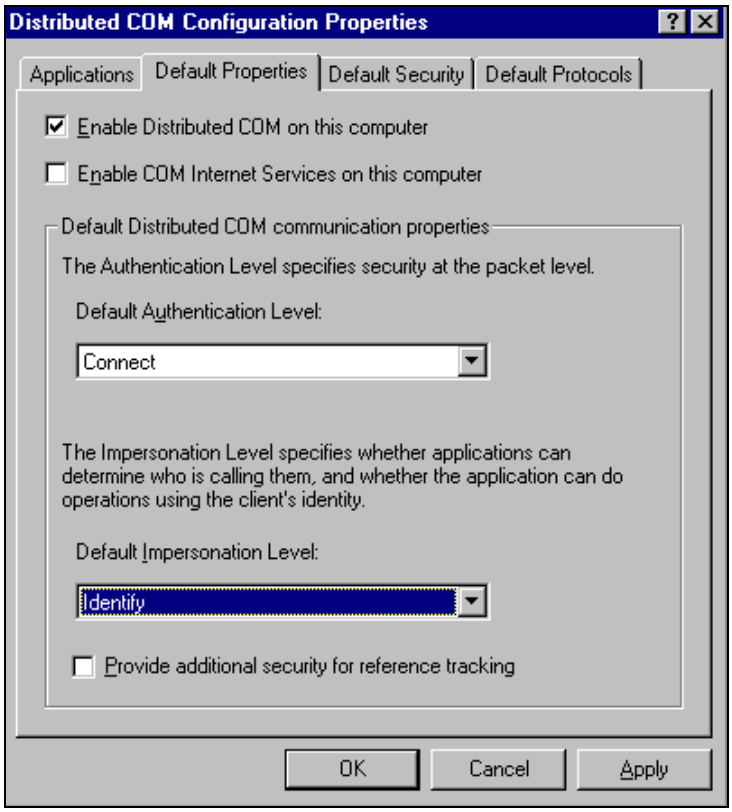


Figure 123: DCOM Default Properties

3. DCOM Configuration Properties-Default Security Tab:

It is on this tab that you can tell the operating system who is allowed to access the applications deploying the OPC AE Controls from remote OPC servers. Default Access Permission is the only setting we are concerned with on the client side of this tab. On the Default Access Permissions Dialog, you will set who (users whose remote OPC servers are running under) will have the ability to make callbacks to this machine when subscription based reads are being done. No changes are normally required on Default Launch Permissions and Default Configuration Permissions dialogs.

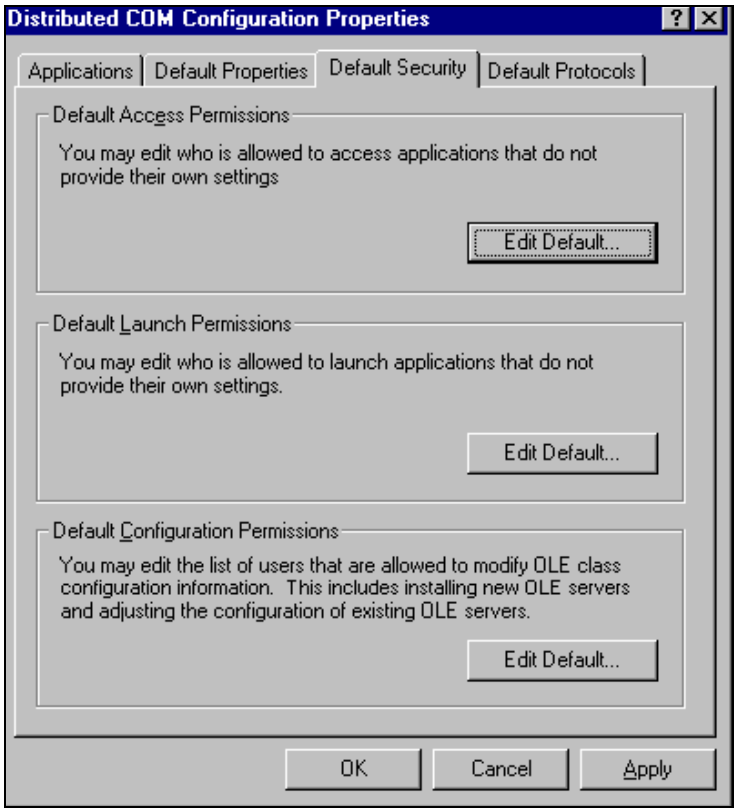


Figure 124:DCOM Default Security Tab

4. DCOM Configuration Properties-Default Protocols Tab:

On this tab you set which of the installed network protocols to use for DCOM on the client computer. You should use Connection-oriented TCP/IP.

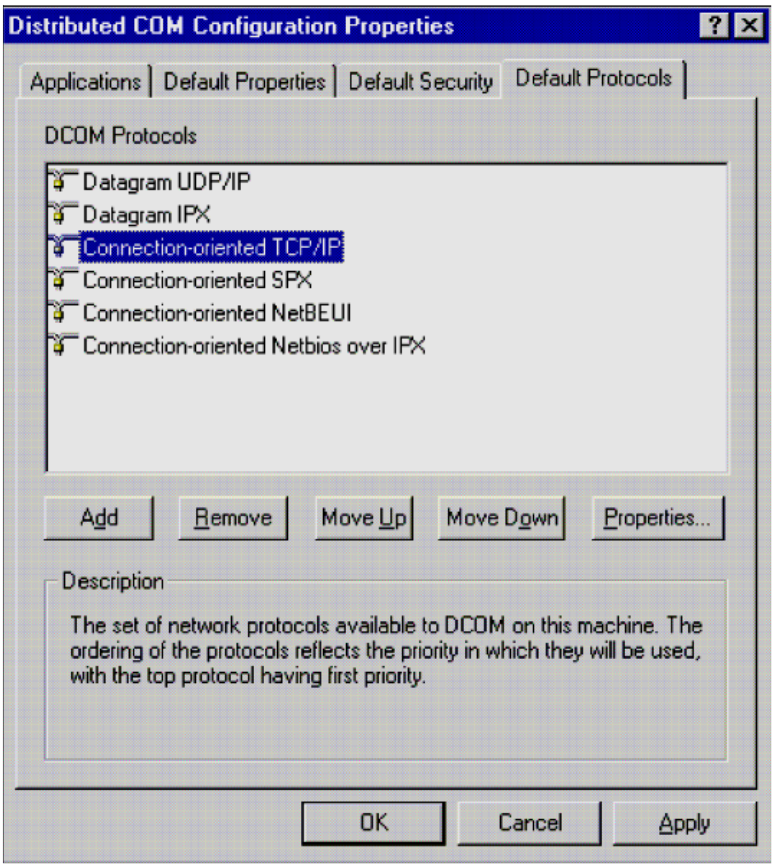


Figure 125 : DCOM Default Protocols

Step 2: You need to register your OPC Server on the client computer by indicating its location on a named remote machine.

Depending on the client environment, there are two ways of registering your OPC server on your client machine. Here are two methods:

1. Prepare and apply a customized .reg file on the client computer (See Microsoft registry documentation for details). We recommend this method only for experienced users with Windows Registry. You have to export the entries of your OPC server from the server machine registry to the client machine registry.

2. Alternatively, install and configure your OPC Server on the client computer. This action self-registers the server in the System Registry. This is the easiest way for an automatic registration.

Then, use the following steps to verify that the OPC server machine is properly delegated:

a. On the client machine, run the DCOM Config Utility (Dcomcnfg.exe).
b. Select your OPC server from the Applications tab and choose Properties.
c. On the General tab, be sure that there is an entry for Remote Computer and that the remote computer name is correct.
d. If the computer name is incorrect, select the Location tab.

e. Make sure the Run application on the following computer setting is checked. In the Dialog box beneath this selection, type in the correct computer name for your OPC server (see the figure below).

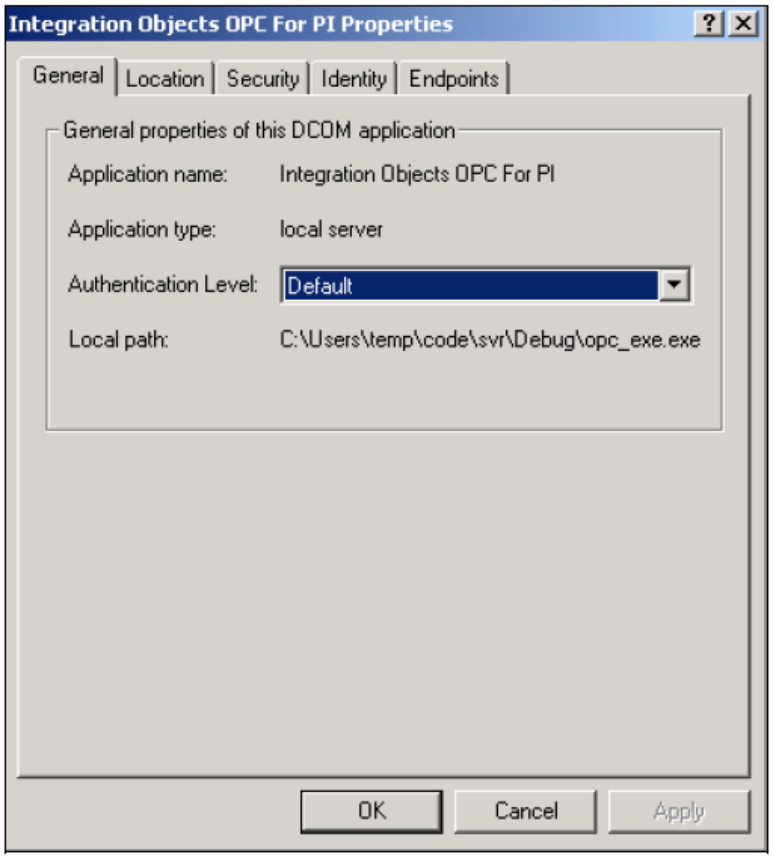


Figure 126: General properties of the selected OPC server

You can also use the following steps to verify the remote computer name by using the Windows Registry:

1. Run RegEdit.exe.
2. The remote server name is specified in the following registry key:
3. HKEY_CLASSES_ROOT\AppID\{The CLSID of the OPC server}\ RemoteServerName

10. Server Side DCOM Configuration

There are 2 areas you will need to setup:

Step 1: Follow these instructions to make default DCOM Configuration for your OPC Server Computer.

1. Launch the DCOM Config Utility on the computer your target OPC Server is running.
2. Configure the Default Properties Tab as you did on the Client side.
3. DCOM Configuration Properties-Default Security Tab:

This tab has the most settings to be made. On this tab, specify for the operating system who is allowed to access OPC servers on this machine (Default Access Permissions), who is allowed to launch OPC Servers on this machine (Default Launch Permissions), and who is allowed to configure OPC Servers on this machine (Default Configuration Permissions).

4. DCOM Configuration Properties-Default Security Tab-Default Access Permissions Dialog:

In the grant access to users and groups, click *Add* on the right of the dialog and you will be presented with another dialog that allows you to browse and select users on the local machine and domain (if applicable and logged into a domain).

5. DCOM Configuration Properties-Default Security Tab-Default Launch Permissions Dialog:

Here, you can define who can actually start your OPC server on this computer. Adding of users/groups is done the same way as was done for Access Permissions.

6. DCOM Configuration Properties-Default Security Tab-Default Configuration Permissions Dialog: If you are setting up DCOM for the first time, it is not recommended to change the settings.
7. Configure the Default Protocols Tab as you did on the Client side.

Step 2: To make DCOM settings that are specific to your OPC Server, go to the Application Tab in DCOM Config and browse until you find the OPC Server of your choice. Highlight it and either double click it or click the Properties to enter the server specific settings.

1. On the General Tab, we recommend that you leave the Authentication Level as **Default**.

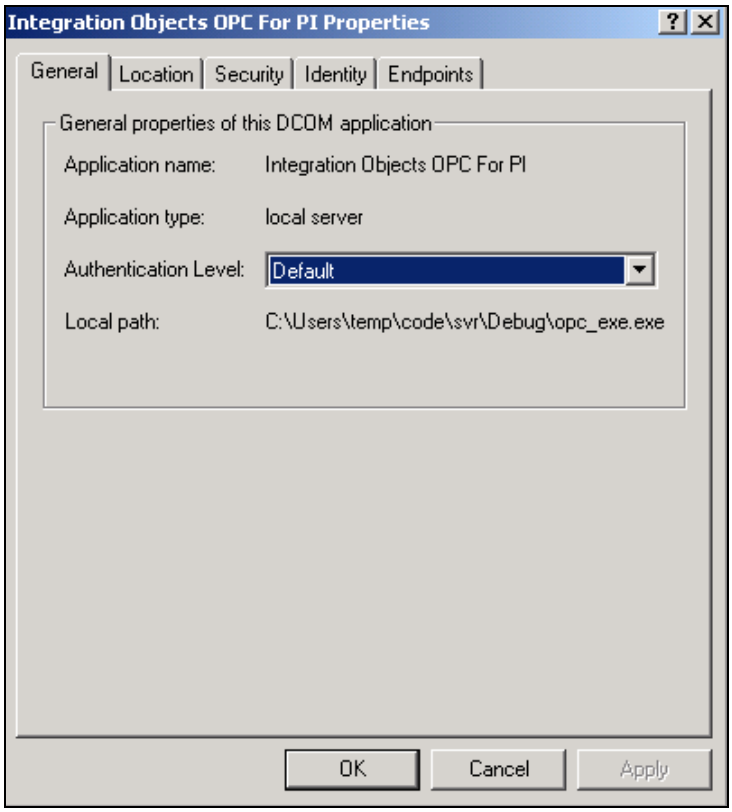


Figure 127 :DCOM Server Configuration

2. On the Location Tab, make sure that the *Run Application* on this computer is the ONLY box that is checked.

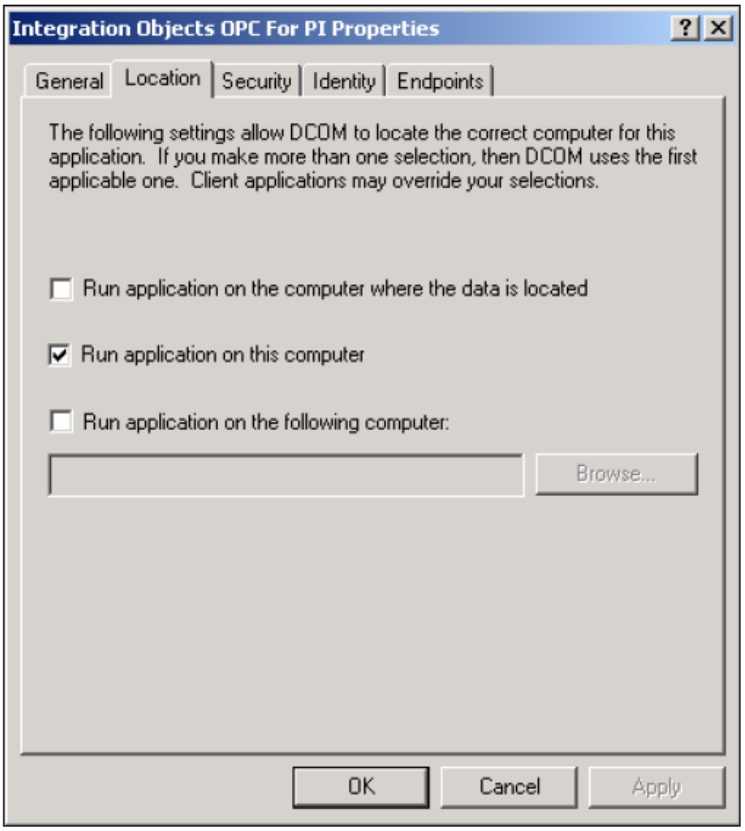


Figure 128:Location configuration of the selected OPC server

3. On the Security Tab, we suggest that you select "Use Default access permissions". This means that users/groups shown under the *Default Security Tab* in the DCOM Config utility will have access to connect to this specific OPC server. If you choose to use the custom permissions to override the defaults, you must specify which users/groups you wish to grant access to. We also suggest that you use the Default Launch permissions. The same rules apply about using custom launch permissions here as they do for custom access permissions.

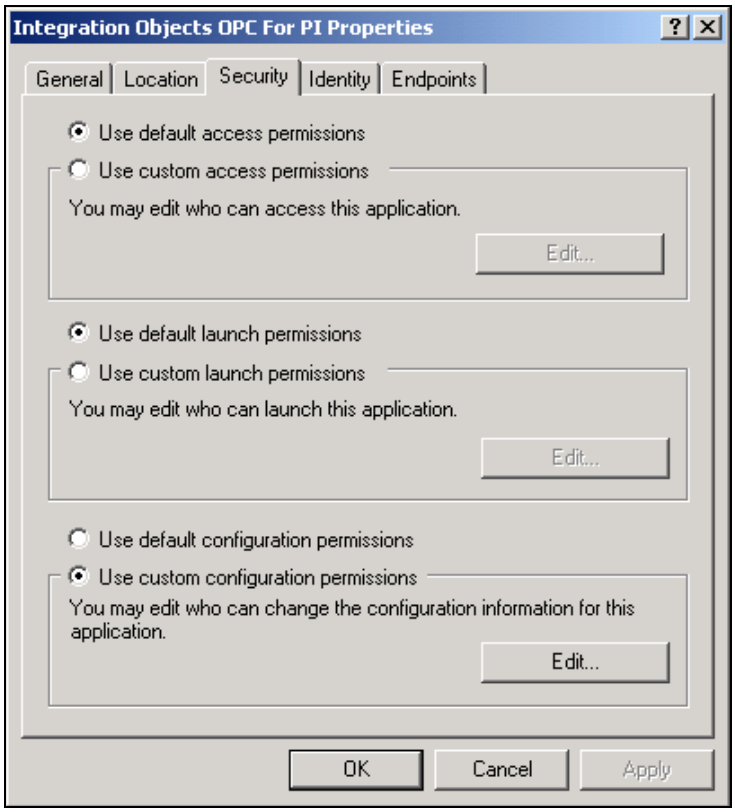


Figure 129: Configuration of DCOM security properties

4. On the *Identity* Tab, specify under which user account you want the OPC server to run. For some cases, this is one of the most important settings for the OPC server. The answer is very dependent on how you will be using your system.
5. No changes are required on the *Endpoints* Tab.

For additional information on this guide, questions or problems to report, please contact:

Offices
- Americas: +1 713 609 9208
- Europe-Africa-Middle East: +216 71 195 360

Email
- Support Services: customerservice@integrationobjects.com
- Sales: sales@integrationobjects.com

To find out how you can benefit from other Integration Objects products and custom-designed solutions, please visit us on the Internet:

Online
- www.integrationobjects.com