Contents

INTRODUCTION........................................................................................................... 4
Conventions used in this documentation................................................................. 5
About this software.................................................................................................. 7
About the BLUE Open Studio software components........................................... 10
Install the full BLUE Open Studio software......................................................... 11
Execution Modes.................................................................................................... 15

THE DEVELOPMENT ENVIRONMENT..................................................................... 17
Title Bar.................................................................................................................... 18
Status Bar............................................................................................................... 19
Application button................................................................................................. 20
Quick Access Toolbar............................................................................................ 21
Ribbon....................................................................................................................... 23
    Home tab.............................................................................................................. 23
    View tab................................................................................................................ 23
    Insert tab.............................................................................................................. 24
    Project tab............................................................................................................ 24
    Graphics tab........................................................................................................ 24
    Format tab............................................................................................................ 25
    Help tab............................................................................................................... 26
Project Explorer..................................................................................................... 27
    Global tab............................................................................................................ 27
    Graphics tab........................................................................................................ 28
    Tasks tab.............................................................................................................. 29
    Comm tab............................................................................................................ 30
Screen/Worksheet Editor....................................................................................... 31

ABOUT TAGS AND THE PROJECT DATABASE......................................................... 32
Understanding the Tag Name Syntax..................................................................... 33
Choosing the Tag Data Type.................................................................................... 34
Using Array Tags..................................................................................................... 35
About indirect tags................................................................................................ 38

TUTORIAL: BUILDING A SIMPLE PROJECT............................................................. 39
Creating a new project............................................................................................ 40
Specifying the startup screen................................................................................ 41
Creating tags............................................................................................................ 42
Creating the main screen....................................................................................... 43
    Drawing the main screen's title........................................................................ 44
    Drawing a button to open another screen......................................................... 45
    Saving and closing the main screen.................................................................. 46

Saving and closing the main screen..................................................................... 46
Drawing a button to open another screen............................................................. 45
Saving and closing the main screen..................................................................... 46
Contents

Creating the synoptic screen...................................................................................................................................................... 47
  Drawing the synoptic screen’s title.................................................................................................................................... 47
  Drawing “Date” and “Time” displays.................................................................................................................................. 47
  Placing an “Exit” icon......................................................................................................................................................... 49
  Testing the project................................................................................................................................................................ 50
  Placing an animated tank...................................................................................................................................................... 50
  Placing a level slider.......................................................................................................................................................... 52
  Drawing a tank selector....................................................................................................................................................... 53
  Testing the project................................................................................................................................................................ 53
Configuring the communication driver........................................................................................................................................ 55
  Monitoring device I/O during runtime.................................................................................................................................. 57
BLUE Open Studio (or BOS, for short) is a powerful, integrated tool that exploits key features of Microsoft operating systems and enables you to build full-featured SCADA (Supervisory Control and Data Acquisition) or HMI (Human-Machine Interface) programs for your industrial automation business.

This BLUE Open Studio Quick Start Guide is intended for individuals using BOS for the first time. This publication will help you quickly familiarize yourself with the basic functions of BOS.
Conventions used in this documentation

This documentation uses standardized formatting and terminology to make it easier for all users to understand.

Text conventions

This documentation uses special text formatting to help you quickly identify certain items:

• Titles, labels, new terms, and messages are indicated using italic text (for example, Object Properties).
• File names, screen text, and text you must enter are indicated using monospace text (for example, D:\Setup.exe).
• Buttons, menu options, and keyboard keys are indicated using a bold typeface (for example, File menu).

In addition, this documentation segregates some text into Tip, Note, and Caution boxes:

• Tips provide useful information to save development time or to improve the project performance.
• Notes provide extra information that may make it easier to understand the nearby text, usually the text just before the note.
• Cautions provide information necessary to prevent errors that can cause problems when running the project, and may result in damage.

Mouse and selection conventions

Because most PCs used for project development run a version of Microsoft Windows with a mouse, this documentation assumes you are using a mouse. Generally, a PC mouse is configured for right-handed use, so that the left mouse button is the primary button and the right mouse button is the secondary button.

This documentation uses the following mouse and selection conventions:

• Click and Select both mean to click once on an item with the left mouse button. In general, you click buttons and you select from menus and lists.
• Double-click means to quickly click twice on an item with the left mouse button.
• Right-click means to click once on an item with the right mouse button.
• Select also means you should use your pointing device to highlight or specify an item on the computer screen. Selecting an item with a touchscreen is usually the same as selecting with a mouse, except that you use your finger to touch (select) a screen object or section. To select items with your keyboard, you typically use the Tab key to move around options, the Enter key to open menus, and the Alt key with a letter key to select an object that has an underlined letter.
• Drag means to press down the appropriate mouse button and move the mouse before releasing the button. Usually an outline of the item will move with the mouse cursor.

Windows conventions

This documentation uses the following Windows conventions:

• Dialogs are windows that allow you to configure settings and enter information.
• Text boxes are areas in dialogs where you can type text.
• Radio buttons are white circles in which a black dot appears or disappears when you click on the button. Typically, the dot indicates the option is selected or enabled. No dot indicates the option is cleared or disabled.
• Check boxes are white squares in which a check (✓) appears or disappears when you click on it with the cursor. Typically, a check ✓ indicates the option is selected or enabled. No check x indicates the option is cleared or disabled.
• Buttons are icons in boxes appear "pressed" when you click on them.
• Lists are panes (white boxes) in windows or dialogs containing two or more selectable options.
• Combo boxes have arrows that, when clicked, show part or all of an otherwise concealed list.
• **Dockable windows** are windows that you can drag to an edge of the interface and merge with that edge.
About this software

BLUE Open Studio is powerful software for developing HMI, SCADA, and OEE/Dashboard projects that can be deployed anywhere.

Each BOS project consists of:

• A project tags database to manage all run-time data, including both internal variables and I/O data;
• Configurable drivers to communicate in real-time with programmable logic controllers (PLCs), remote I/O devices, and other data-acquisition equipment;
• Animated HMI screens and OEE dashboards; and
• Optional modules such as alarms, events, trends, recipes, reports, scriptable logic, schedulers, a project security system, and a complete database interface.

After you develop your project, you can either run it locally on your development workstation or download it to a remote computer and run it there. The project runtime server processes I/O data from connected devices according to your project parameters and then reacts to, displays, and/or saves the data.

Product features

ActiveX and .NET

Use third-party controls to enhance your project. BOS is a container for ActiveX and .NET controls. Add functionality such as browsers, media players, charting, and other tools that support the ActiveX and .NET interface standards.

Alarms

In addition to all the alarm functions you’d expect, BOS v8.0 also sends alarms using multi-media formats like PDF. Use remote notification to have alarms sent right to your inbox, a printer, or a smartphone! Alarms are real-time and historical, log data in binary format or to any database.

Animation

BOS gives you great command over graphics. Paste images, and even rotate them dynamically. Fill bar graphs with color, or adjust the scale of objects with easy-to-use configuration. Other animations include “command” (for touch, keyboard and mouse interaction), hyperlink, text data link, color, resize (independent height and width), position, and rotation (with custom rotation point).

Drivers

BOS v8.0 contains over 240 built-in drivers for most PLCs, temperature controllers, motion controllers, and bar code/2D/RFID readers. Digital Electronics Corporation driver toolkits allow you the flexibility to build your own drivers. Use these built in drivers without the need for OPC servers (but are an optional connection method).

Email

Send email using SMTP to desktop, email enabled phone, or any enabled device. Get real-time information on alarms, process values, and other events. BLUE Open Studio v8.0 supports SSL encryption allowing the use of third-party providers such as Gmail.

Events

BOS v8.0 offers traceability for operator initiated actions or internal system activity. Log events such as security system changes (user logon or off), screen open/close, recipe/report operations, custom messages and system warnings. Also any tag value changes including custom messages.

FDA Traceability

Take advantage of built-in functionality to create 21 CFR part 11 compliant projects with traceability and e-signatures. These features are often used for pharmaceutical and food applications, but also for any application where traceability is a must.
FTP
Automatically upload or download files during runtime to/from remote storage locations using FTP protocol and flexible scripting functions. Configure FTP via scripting or the included configuration interface.

Graphics and Design Tools
Create powerful screens to meet any application need using the improved tools in our graphic interface. Combine built-in objects to create any functionality required. Store graphics in the library for future use, or easily make project across a product line share a consistent "look and feel".

Historical Performance
We have optimized the trend history module, and designed it to load millions of values from SQL Relational Databases with high performance, with built-in data decimation in the Trend Control. Easy to use tools provide quick access to Statistical Process Control (SPC) values without any need for programming.

Intellectual Property Protection
Screens, documents, scripts and even math worksheets can be individually password protected. This prevents unauthorized viewing or editing of your corporate custom functionality. Protect the entire project with just a few mouse clicks.

Multi-Language
Develop your project in one of many development languages, including English, Portuguese, German, and French.

OPC
As an alternative to the built-in drivers for direct communication with PLCs, you can also use any of several different versions of OPC to manage your devices. BLUE Open Studio includes support for "classic" OPC DA (server or client), OPC HDA (server), OPC UA (client), OPC .NET (client), and OPC XML-DA (client).

PDF Export
Send Alarms, Reports, or any file (including .doc or .txt) to a production supervisor, quality manager, or maintenance staff using the included PDF writer.

Recipes
Save time and maintain consistency by automating part parameters or productions quantities with any triggering event.

Redundancy
For critical applications where data is vital, BOS v8.0 supports web server, database and overall system redundancy.

Reports
Create clear, concise reports in text format, graphical RTF, XML, PDF, HTML, and CSV or integrate with Microsoft Office. Get the data you need, in the format you need it, to make informed decisions, fast.

Scalable
Develop once and deploy anywhere, on any currently supported version of Microsoft Windows.

Scheduler
Schedule custom tag changes on date/time, frequency, or any trigger. Use this for simulation, to trigger reports or other functionality at a particular time of day, or even to trigger driver worksheets to read/write at a scan rate you choose.

Scripting
Two powerful scripting languages are supported. Use built-in functions or use standard VBScript to take advantage of widely available resources. Both can be used simultaneously to give you the functionality you need.

**Security**

BOS includes support for group and user accounts, e-signatures, and traceability, as well as support for the ADAM Server, in addition to standard LDAP Servers. Integrate your project to the Active Directory (Users and Groups).

**SSL Support for Emails**

Native support for Secure Socket Layer (SSL), makes it easy and secure to send emails from BLUE Open Studio using third-party tools such as Gmail!

**Standards**

Take advantage of common industry standards to develop projects that are compatible with any format. TCP/IP, .Net, ActiveX, OPC (client and server), ADO/ODBC, COM/DCOM, OLE, DDE, XML, SOAP, and HTML are supported.

**SNMP**

Easily configure managed networked devices on IP networks (such as switches and routers) using incorporated SNMP configuration commands and an easy-to-use configuration interface.

**Symbols**

Included library features push buttons, pilot lights, tanks, sliders, meters, motors, pipes, valves and other common objects. Use the included symbols in your project, modify existing symbols to suit your needs, or create your own from scratch. Plus support for third-party symbol libraries and graphic tools.

**Tag Database**

BOS features an object oriented database with boolean, integer, real, strings, arrays, classes (structures), indirect tags and included system tags.

**Trends**

Real-time and Historical trends are supported. Log data in binary format or to any database locally and remotely. Color or fill trends with graphic elements to enhance clarity of data. Date/Time based or numeric (X/Y plot) trends give you the flexibility to display information that best suits your project.

**Troubleshooting**

Quickly debug and verify a project using local and remote tools for troubleshooting, including status fields, DatabaseSpy and LogWin. Capture screen open and close times, see communications in real-time, and messages related to OPC, recipes/reports, security, database errors and even custom messages. Quickly get your project finished using these powerful tools.
About the BLUE Open Studio software components

The BLUE Open Studio software suite comprises several individual components that can be installed on different platforms to perform different functions. The architecture of your finished BOS project depends on which components you install, where you install them, and how you connect them to each other.

The following table lists all of the available components.

<table>
<thead>
<tr>
<th>Component</th>
<th>Features</th>
<th>Platforms</th>
</tr>
</thead>
</table>
| BLUE Open Studio         | • Project development environment  
                           | • Tag integration              | • Windows                        |
|                          | • Remote management of project runtimes                                | • Windows Server                 |
|                          | • Project runtime                                                       | • Windows Embedded Standard      |
|                          | • Agent to allow remote management                                      |                                  |
|                          | • Project thin client                                                   |                                  |
| Mobile Access Runtime    | Enables the project runtime to serve HTML5-enhanced project screens to  | • Internet Information Services (IIS) for Windows |
|                          |   tablets and smartphones.                                              | • any CGI-enabled web server (e.g., Apache) |

It is important to distinguish between the project development environment and the project runtime. The project development environment enables you to design, develop, troubleshoot, deploy, and maintain BOS projects. The project runtime actually runs your project, communicates with external databases and devices, and serves project screens to thin clients.

The full BLUE Open Studio software includes both the project development environment and the project runtime. Your software license determines which parts of the software you can use. For more information, see Execution Modes on page 15.

In most cases, the first thing you should do is install the full BLUE Open Studio software on your primary workstation, because it not only sets up the project development environment for you, it also unpacks the rest of the components so that they can be installed on other computers and devices.
Install the full BLUE Open Studio software

Install the full BLUE Open Studio software on your Windows computer in order to develop BOS projects, or to use the computer as a project runtime server.

To install and run the full BLUE Open Studio software, you must have:

- A Windows-compatible computer with a standard keyboard, a pointer input (i.e., a mouse, trackpad, or touchscreen), and an SVGA-minimum display;
- One of the following Windows operating systems:
  - Windows 7 Service Pack 1
  - Windows 8.1
  - Windows 10
  - Windows Server 2008 R2 Service Pack 1
  - Windows Server 2012 R2
  - Windows Server 2016
  - Windows Embedded 7 Standard
- .NET Framework 3.5 (see note below);
- Microsoft Internet Explorer 6.0 or later;
- 2 GB free storage (hard drive or non-volatile);
- 1 GB free memory (RAM); and
- An Ethernet or Wi-Fi network adapter.

We recommend the Home Premium, Professional, Enterprise, and Ultimate editions of Windows, because they include Internet Information Services (IIS) as a pre-installed feature that can be turned on. You can use IIS to make your projects accessible to thin clients and mobile devices. We do not recommend the Starter and Home Basic editions because they do not include IIS, but you can still use them if you do not plan to use web-based features.

Only Windows 8.1, Windows 10, Windows Server 2012 R2, and Windows Server 2016 are under what Microsoft calls "mainstream support", which means they are actively maintained and additional service packs might be released for them in the future. Windows 7 and Windows Server 2008 R2 are under what Microsoft calls "extended support", which means they are no longer actively maintained. For more information, go to: windows.microsoft.com/en-us/windows/lifecycle

The following items are optional but recommended:

- A DVD-ROM drive, to install the software from an installation disc.
  This is optional because you can also download the installer over the network to your computer.
- Serial COM ports and adapters, to be used for direct communication with PLCs and other devices.
  This is optional because many newer device protocols use TCP/IP or UDP/IP communication (i.e., Ethernet) instead of serial communication.
- Internet Information Services (IIS) installed and turned on, to make your projects accessible to mobile devices. For more information, see the description of the Mobile Access Runtime feature below.
  This is optional because you may choose not to install the Mobile Access Runtime feature now, as part of the full BLUE Open Studio software. You can install it later, for either IIS or CGI.
- Microsoft Visual Studio 2010 or Team Explorer 2010 installed, to enable workgroup collaboration and source control. For more information, see the description of the Collaboration feature below.
  This is optional because you may choose not to install the Collaboration feature.

Finally, you must have Administrator privileges on the computer in order to install software.

Note: .NET Framework 3.5 is pre-installed in most recent versions of Windows, but it might not be turned on by default. To turn it on, use the Programs and Features control panel.
In older versions of Windows, .NET Framework 3.5 must be installed separately. The BLUE Open Studio software installer will attempt to do this for you (see Step 2 below), but depending on your computer’s security settings, the installation might fail without notice. If you experience problems later, while trying to run BLUE Open Studio, use the Add/Remove Programs control panel to confirm that .NET Framework 3.5 was installed successfully.

In Windows Server 2012 R2, .NET Framework 3.5 is not pre-installed and it cannot be installed by the BLUE Open Studio software installer. You must use the Server Manager utility to install it. For more information, see Install .NET Framework 3.5 in Windows Server 2012 R2.

Later versions of .NET Framework (e.g., .NET Framework 4.5) do not include .NET Framework 3.5.

To install the full BLUE Open Studio software:

1. Do one of the following:

   • Download the zipped installer to your computer, either from our website (www.pro-face.com/trans/en/product/1041.html) or from another location on your network where you have previously saved it. Extract the files, open the resulting folder, and then locate and run the setup program (setup.exe).

   • Insert the installation disc into your DVD-ROM drive. If it does not autorun, locate and open the Welcome page (D:\proface.htm). When the page is opened in your browser, click Product Installation and then follow the instructions.

   The installation wizard runs and asks you to select a language for the installation.

2. Select a language from the list, and then click OK.

   This selection determines the language of the user interface for both the installation wizard and the project development environment. You can change the language for the project development environment later, after the software has been installed.

   The wizard prepares for installation. During this step, it automatically installs SafeNet’s Sentinel drivers (a part of the software licensing mechanism) and .NET Framework 3.5.

3. On the Welcome page of the wizard, click Next to proceed with the installation.

4. On the License Agreement page, click Yes to accept the agreement and proceed, or click No to refuse the agreement and exit the wizard.

5. On the Customer Information page, type your user name and company name, and then click Next.

6. On the Choose Destination Location page, select the folder where the software should be installed, and then click Next.

   By default, the software will be installed at:

   C:\Program Files\BLUE Open Studio v8.0\BLUE Open Studio v8.0\

7. On the Select Features page, select the specific features and components that you want to install, and then click Next.

<table>
<thead>
<tr>
<th>Feature</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Program Files</td>
<td>The main program files for the project development application, the project runtime server, and the project runtime client. This feature cannot be deselected.</td>
</tr>
<tr>
<td>Demos</td>
<td>Premade projects that demonstrate the capabilities of the BLUE Open Studio software.</td>
</tr>
<tr>
<td>OPC Components</td>
<td>Additional components required for communication with other OPC-compatible devices. This includes OPC DA (a.k.a. OPC Classic), OPC UA, OPC .NET (a.k.a. OPC Xi), and OPC XML-DA.</td>
</tr>
<tr>
<td>PDF Printing</td>
<td>Additional software that allows run-time reports to be saved as PDF files.</td>
</tr>
<tr>
<td>Security System Device Driver</td>
<td>An additional keyboard driver that enforces project security during run time by controlling user input.</td>
</tr>
</tbody>
</table>
Feature | Description
--- | ---
Symbol Library | A library of premade but configurable screen objects such as pushbuttons, toggle switches, gauges, dials, indicator lights, and so on.
Mobile Access Runtime | Additional software for Internet Information Services (IIS) that makes your project runtime accessible to mobile devices such as tablets and smartphones.

This feature requires that you have IIS turned on and configured with ASP, ASP.NET, and ISAPI Extensions enabled. The BLUE Open Studio software installer will attempt to verify that you do, and if you do not, it will not install this feature.

For more information, see Turn on IIS for thin client access.

You do not need to install this feature at this time. You can install it later, after you have turned on IIS, or you can install it on another computer that is acting as your project runtime server. There is a separate Mobile Access Runtime software installer (MobileAccessSetup.exe) that is unpacked with the rest of the BLUE Open Studio software.

To use this feature, your software license must include the Mobile Access Runtime option. For more information, see About license settings. To purchase the option, contact your software distributor. You may still develop projects that include Mobile Access features, even without the option, but clients will not be able to access them during run time.

Collaboration | Additional tools for workgroup collaboration and source control within the BLUE Open Studio project development environment.

This feature requires that you have Microsoft Visual Studio Team Explorer 2010 installed on the same computer. The BLUE Open Studio software installer will attempt to verify that you do, and if you do not, it will not install this feature.

The Team Explorer module is included in some versions of Microsoft Visual Studio 2010, so if you already have Visual Studio installed on your computer, you might be able to select and install the Collaboration feature. However, if you do not have Visual Studio installed, or if you are not sure that your version of Visual Studio includes the Team Explorer module, you can separately download and install Team Explorer 2010 for free.


*Note:* Team Explorer 2012 and Team Explorer 2013 are not supported as collaboration clients at this time.

You should also have Microsoft Visual Studio Team Foundation Server 2010 or 2012 running somewhere on your network, but if you do
Feature | Description
--- | ---
| | not, it will not prevent you from installing the Collaboration feature now.

**Note:** Team Foundation Server 2013 is not supported at this time for the collaboration server.

To use this feature, your software license must include the Collaboration option. For more information, see About license settings. To purchase the option, contact your software distributor.

8. On the **Ready To Install** page, click **Install**.

**Note:** You might receive the following error message during installation: "Error 1628: Failed to complete script based install." For more information about this error and how to resolve it, go to: flexeracommunity.force.com/customer/articles/en_US/ERRDOC/Error-1628-Failed-To-Complete-Script-Based-Install

The software is installed, and then when the installation is finished, the last page of the wizard is displayed.

9. Click **Finish** to close the installation wizard.

When you have finished the installation, you should find the BLUE Open Studio software in your Windows Start menu at **Start > All Apps > BLUE Open Studio v8.0**. It includes the following components:

**BOS v8.0 BLUE Open Studio**
The project development environment, runtime.

**BOS v8.0 Help Manual**
A complete technical reference and user guide for all of BLUE Open Studio.

**BOS v8.0 Quick Start Guide**
A brief guide to installing and using the project development environment, including a tutorial for developing a simple project.

**BOS v8.0 Register**
A utility program that manages your BLUE Open Studio software license.

**BOS v8.0 Release Notes**
A list of changes in BLUE Open Studio.

**BOS v8.0 Remote Agent**
A utility program that allows BLUE Open Studio running on other computers to connect to your computer and send projects to it.

**BOS v8.0 StartUp**
A shortcut that automatically starts the project runtime and runs the most recent project.

There should also be a shortcut icon on your desktop.

To run the software, do one of the following:

- Double-click the shortcut icon on your desktop; or
- Click **Start > All Apps > BLUE Open Studio v8.0 > BOS v8.0 BLUE Open Studio.**
Execution Modes

BLUE Open Studio, EmbeddedView, and CEView support the following execution modes:

<table>
<thead>
<tr>
<th>Execution Mode</th>
<th>BLUE Open Studio</th>
<th>EmbeddedView / CEView</th>
</tr>
</thead>
<tbody>
<tr>
<td>Evaluation Mode</td>
<td>#</td>
<td></td>
</tr>
<tr>
<td>Demo Mode</td>
<td>#</td>
<td>#</td>
</tr>
<tr>
<td>Licensed for Engineering Only</td>
<td>#</td>
<td></td>
</tr>
<tr>
<td>Licensed for Runtime Only</td>
<td>#</td>
<td>#</td>
</tr>
<tr>
<td>Licensed for Engineering + Runtime</td>
<td>#</td>
<td></td>
</tr>
</tbody>
</table>

**Evaluation Mode**

Enables all of the product’s engineering and runtime features.

The first time you install BLUE Open Studio on a computer, the product runs for forty (40) hours in Evaluation Mode. This evaluation period includes any time you run a product module (engineering or runtime). You can use this evaluation period continuously or not; for example, 10 hours a day for 4 days, or 5 hours a day for 8 days, or 10 hours a day for 3 days plus 5 hours a day for 2 days, and so on.

After running for 40 hours in the Evaluation Mode, the evaluation period ends and the program automatically converts to Demo Mode until you apply a valid license. You cannot reactivate Evaluation Mode, even if you reinstall the software on your computer.

**Note:** Each version of BLUE Open Studio has an evaluation period that is independent of every other version. For example, if your BLUE Open Studio v7.0 evaluation period has expired and you are running in Demo Mode because you have not installed a license, when you install BLUE Open Studio v8.0 on the same computer, the newer version will begin its own 40-hour evaluation period and the older version will continue running in Demo Mode.

**Demo Mode**

Allows you to download projects to remote stations and to run projects for testing or demonstration purposes. You can execute runtime tasks and use the debugging tools (*LogWin* and *Database Spy*), but they shut down automatically after running for two hours continuously. You can restart the Demo Mode again and run for another two hours, and so on.

You cannot create or modify screens, worksheets, or project settings in Demo Mode.

**Licensed for Engineering Only**

Enables all development options for an unlimited time.

This mode also allows you to continuously run the runtime tasks and debugging tools (*Database Spy*, *Output* window, and *LogWin* module) for 72 hours. After that period, these tasks shut down, but you can restart them and run for another 72 hours, and so on. You can use this license for development and testing only.

**Licensed for Runtime Only**

Enables all runtime tasks and debugging tools (*Database Spy*, *Output* window, and *LogWin* module) for unlimited time, but you cannot create or modify screens and/or worksheets.

The menu options available in Runtime Only mode are the same as the options listed for Demo Mode (see previous table).

**Licensed for Engineering + Runtime**

Enables all development options, runtime tasks, and debugging tools (*Database Spy*, *Output* window, and *LogWin* module) for an unlimited time.
**Note:** The Remote Management tool is always available, regardless of the execution mode, so that you can upload files from or download files to remote stations.

To see which execution mode you are currently running, click **About** on the Help tab of the ribbon; the **About** dialog shows the execution mode, including the time remaining if you are in Evaluation Mode.
The Development Environment

BLUE Open Studio incorporates a modern, Ribbon-based Windows interface to provide an integrated and user-friendly development environment.
Title Bar

The Title Bar located along the top of the development environment displays the application name (e.g., BLUE Open Studio) followed by the name of the active screen or worksheet (if any).

The Title Bar also provides the following buttons (from left to right):

- **Minimize** button: Click to minimize the development environment window to the Taskbar.
- **Restore Down / Maximize**: Click to toggle the development environment window between two sizes:
  - **Restore Down** button reduces the window to its original (default) size.
  - **Maximize** button enlarges the window to fill your computer screen.
- **Close** button: Click to save the database and then close the development environment. If you modified any screens or worksheets, the application prompts you to save your work. This button's function is similar to clicking **Exit Application** on the Application menu.

Note: Closing the development environment does not close either the project viewer or the runtime system, if they are running.
The Status Bar located along the bottom of the development environment provides information about the active screen (if any) and the state of the application.

<table>
<thead>
<tr>
<th>Field</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Execution Mode</td>
<td>The current execution mode of the application.</td>
</tr>
<tr>
<td>CAP</td>
<td>Indicates whether the keyboard Caps Lock is on (black) or off (grey).</td>
</tr>
<tr>
<td>NUM</td>
<td>Indicates whether the keyboard Num Lock is on (black) or off (grey).</td>
</tr>
<tr>
<td>SCRL</td>
<td>Indicates whether the keyboard Scroll Lock is on (black) or off (grey).</td>
</tr>
<tr>
<td>Object ID</td>
<td>The ID number of a selected screen object.</td>
</tr>
<tr>
<td>Cursor Position</td>
<td>The location of the cursor on the active screen or worksheet. If it’s a screen, the position of the mouse cursor is given as X,Y coordinates, where X is the number of pixels from the left edge of the screen and Y is the number of pixels from the top edge of the screen. If it’s a worksheet, then the position of the text cursor is given as Line and Column.</td>
</tr>
<tr>
<td>Object Size</td>
<td>The size (in pixels) of a selected screen object, where W is the width and H is the height.</td>
</tr>
<tr>
<td>No DRAG</td>
<td>Indicates whether dragging is disabled (No DRAG) or enabled (empty) in the active screen.</td>
</tr>
<tr>
<td>Tag Count</td>
<td>The total number of tags used so far in the project.</td>
</tr>
</tbody>
</table>

Example of Status Bar


The Status Bar fields (from left to right) are described in the following table:
Application button

The Application button opens a menu of standard Windows application commands like New, Open, Save, Print, and Close.

*Application button opens menu of commands*
Quick Access Toolbar

The Quick Access Toolbar is a customizable toolbar that contains a set of commands that are independent of the ribbon tab that is currently displayed.

Move the Quick Access Toolbar

The Quick Access Toolbar can be located in one of two places:

- Upper-left corner next to the Application button (default location); or
- Below the ribbon, where it can run the full length of the application window.

If you don’t want the Quick Access Toolbar to be displayed in its current location, you can move it to the other location:

1. Click Customize Quick Access Toolbar.
2. In the list, click Show Below Ribbon or Show Above Ribbon.

Add a command to the Quick Access Toolbar

You can add a command to the Quick Access Toolbar directly from commands that are displayed on the ribbon:

1. On the ribbon, click the appropriate tab or group to display the command that you want to add to the Quick Access Toolbar.
2. Right-click the command, and then click Add to Quick Access Toolbar on the shortcut menu.

You can also add and remove commands — as well as reset the toolbar to its default — using the Customize dialog:

1. Click Customize Quick Access Toolbar.
2. In the list, click More Commands. The Customize dialog is displayed.

3. In the Choose commands from menu, select the appropriate Ribbon tab. The commands from that tab are displayed in the Commands list.
4. In the Commands list, select the command that you want to add to the Quick Access Toolbar.
5. Click Add.
Only commands can be added to the Quick Access Toolbar. The contents of most lists, such as indent and spacing values and individual styles, which also appear on the ribbon, cannot be added to the Quick Access Toolbar.
Ribbon

The new ribbon combines the numerous menus and toolbars from the previous version of BOS into a single, user-friendly interface. Almost all application commands are now on the ribbon, organized into tabs and groups according to general usage.

The Ribbon interface

Home tab

The Home tab of the ribbon is used to manage your project within the development environment.

Home tab of the ribbon

The tools are organized into the following groups:

• **Clipboard**: Cut, copy, paste, and find items in project screens and task worksheets.

• **Local Management**: Run and stop the project on the local station (i.e., where the development application is installed), as well as manage the execution tasks. You can also run a project in Debug mode, for debugging VBScript.

• **Remote Management**: Connect to a remote station (e.g., a Windows Embedded device) so that you can download the project to it, and then run, stop, and troubleshoot the project on that station. For more information, see About remote management.

• **Tools**: Miscellaneous tools to verify the project, import tags from other projects, convert screen resolutions, and register ActiveX and .NET controls.

• **Tags**: Manipulate tags and tag properties in the project database.

View tab

The View tab of the ribbon is used to customize the look of the development environment itself.

View tab of the ribbon

The tools are organized into the following groups:

• **Show/Hide**: Show and hide the different parts of the development environment, as well as restore the default layout.
• **Zoom**: Zoom in and out of the screen editor.
• **Options**: Change the language and font used in the development environment.
• **Window**: Arrange the windows in the development environment.

**Insert tab**

The *Insert* tab of the ribbon is used to insert new tags, screens, worksheets, and other components into your project.

The tools are organized into the following groups:

• **Global**: Insert tags, classes, translations, and procedures into the *Global* tab of the Project Explorer.
• **Graphics**: Insert screens and screen groups into the *Graphics* tab of the Project Explorer.
• **Task Worksheets**: Insert task worksheets into the *Tasks* tab of the Project Explorer.
• **Communication**: Insert server configurations and communication worksheets into the *Comm* tab of the Project Explorer.

**Project tab**

The *Project* tab of the ribbon is used to configure your project settings.

The tools are organized into the following groups:

• **Settings**: Configure the general project settings, set the project to run as a Windows service, or enable workgroup collaboration and version control.
• **Security System**: Enable and configure the project security system.
• **Web**: Configure the project to accept connections from a variety of thin clients.

**Graphics tab**

The *Graphics* tab of the ribbon is used to draw project screens.
The tools are organized into the following groups:

- **Screen**: Configure settings for the project screen itself, such as its attributes, script, and background color or image.
- **Editing**: Select and edit objects in the project screen.
- **Shapes**: Draw static lines and shapes.
- **Active Objects**: Draw active objects, like buttons and check boxes.
- **Data Objects**: Draw objects that display historical data, like alarms, events, and trends.
- **Libraries**: Select from libraries of premade objects, such as symbols, .NET and ActiveX controls, and external image files.
- **Animations**: Apply animations to other screen objects.

### Format tab

The **Format** tab of the ribbon is used to format and arrange objects in a project screen.

The tools are organized into the following groups:

- **Arrange**: Arrange objects in a project screen, including bring to front and send to back, group, align, and rotate.
- **Position**: Precisely adjust the position of a screen object in a project screen.
- **Size**: Precisely adjust the size of a screen object.
- **Style**: Change the fill and line color of a screen object.
- **Fonts**: Change the caption font of a screen object.
**Help tab**

The Help tab of the ribbon provides additional help with using the software.

![Help tab of the ribbon](image)

The tools are organized into the following groups:

- **Documentation**: Access the documentation for the development application, including this help file / technical reference and notes for the individual communication drivers.

- **Information**: Access other information about BLUE Open Studio, including the license agreement, product website, and release notes, as well as system and support details that make it easier for Customer Support to assist you.
Project Explorer

The Project Explorer organizes all of the screens, worksheets, and other items that comprise your project and presents them in an expandable tree-view.

To open a folder and view its contents, either click the Expand icon to the left of the folder or double-click the folder itself.

To close a folder, click the Collapse icon to the left of the folder.

If you right-click any item in the Project Explorer, then a shortcut menu will appear with contextual commands for that item.

There are four main sections, or tabs, in the Project Explorer: Global, Graphics, Tasks, and Comm.

**Global tab**

The Global tab of the Project Explorer contains the project tags database, as well as other features that apply to the entire project such as the security system, VBScript procedures, and UI translation.

The folders on the Global tab are described in the following sections:

**Project Tags**

The project tags database contains all of the data tags that you create during project development, such as screen tags (e.g., `button1_state`) or tags that read from / write to connected devices.

**Classes**

Classes are compound tags that you can create to associate a set of values, rather than a single value, with an object. For example, where you may normally create separate tags for a tank's pressure, its temperature, and its fill level, you can instead create a “tank” class that includes all three.

**Shared Database**

The shared database contains tags that were created in another program and then imported into or integrated with your project.

**System Tags**

System tags are predefined values such as the date, the time, the name of the current user, and so on. You can use these values to develop supervisory functions and housekeeping routines.

All system tags are read-only, which means you cannot add, edit, or remove these tags from the database.

**Security**

If you choose to enable it, you can use the project security system to control who may log on to your project and what they may do during runtime.

**Procedures**

Procedures are VBScript functions and sub-routines that can be called by any other script in your project.
**Event Logger**
The event logger saves important runtime messages and task results to an external database.

**Translation**
You can use the translation table to develop a multilingual user interface (MUI) for your project.

**Graphics tab**
The Graphics tab of the Project Explorer contains all of the screens, screen groups, and symbols in your project.

The folders on the Graphics tab are described in the following sections:

**Screens**
You create screens to provide a graphical interface for your project. Each screen can contain many buttons, sliders, dials, indicators, graphs, and so on.

**Screen Groups**
You can combine individual screens into screen groups, so that they all open together at the same time.

**Thin Clients**
You can deploy your project as a web application to be accessed by thin clients such as desktop web browsers, tablets, and smartphones. You can even deploy different versions of your project with different levels of functionality for each type of client.

**Project Symbols**
This folder contains all of the custom symbols that you create for your project. A symbol is a group of interconnected screen objects that work together to perform a single function — for example, lines, rectangles, and text fragments that have been arranged to make a slider control.

**Graphics Script**
You can use this worksheet to define VBScript sub-routines that are called only when the graphics module starts (i.e., when a client station connects to the server and displays the graphical interface), while it is running, and when it ends.

**Symbols**
The symbols library contains not only the custom symbols that you create (see Project Symbols above), but also a large selection of premade symbols that are installed with the development application.

**Layout**
The layout editor displays all of the screens that are currently open for editing. You can use it to visualize how the screens are arranged together and reuse screens in multiple layouts — for example, to create a common navigation bar across your entire project.
The Development Environment

Tasks tab
The Tasks tab of the Project Explorer organizes the worksheets that are processed as background tasks (i.e., server-based maintenance tasks that are not directly related to screen operations or device I/O) during project runtime.

![Tasks tab of the Project Explorer](image)

The folders on the Tasks tab are described in the following sections:

**Alarms**
You can use Alarm worksheets to define when alarms are triggered, how they must be handled, and what messages they generate. (You can then use the Alarm/Event Control screen object to display your alarms on screen, but that is a separate procedure.)

**Trends**
You can use Trend worksheets to select project tags that should be displayed as data trends and/or saved as historical data. (You can then use the Trend Control screen object to actually display your trends on screen, but that is a separate procedure.)

**Recipes**
You can use Recipe worksheets to select project tags that will load values from and/or save values to an external file. These worksheets are typically used to execute process recipes, but you can store any type of information such as passwords, operation logs, and so on. (You can then call the `Recipe` function to actually run a configured Recipe worksheet, but that is a separate procedure.)

**Reports**
You can use Report worksheets to design runtime reports that are either sent to a printer or saved to disk. (You can then call the `Report` function to actually run a configured Report worksheet, but that is a separate procedure.)

**ODBC**
You can use ODBC worksheets to set up connections and exchange data with other ODBC-compliant databases.

**Math**
You can use Math worksheets to develop complex runtime logic using the built-in scripting language.

**Script**
You can use Script worksheets to develop complex runtime logic using VBScript.

**Scheduler**
You can use Scheduler worksheets to run commands at specified times, dates, or trigger events.

**Database**
You can use Database worksheets to set up connections and exchange data with external databases using the standard ADO.NET interface (as an alternative to ODBC).
Comm tab

The Comm tab of the Project Explorer organizes the worksheets that control communication with remote devices, using either direct communication drivers or other common protocols.

The folders on the Comm tab are described in the following sections:

Drivers
You can use Driver worksheets to communicate with PLCs and other hardware, using any of the hundreds of direct communication drivers that are installed with the development application.

OPC DA 2.05
You can use OPC worksheets to communicate with OPC servers via the OPC Classic protocol.

OPC UA
You can use OPC UA worksheets to communicate with OPC servers via the new OPC Unified Architecture protocol.

OPC .Net
You can use OPC .Net worksheets to communicate with OPC servers via the new OPC .NET 3.0 protocol (formerly OPC Xi).

OPC XML/DA
You can use OPC XML/DA worksheets to communicate with OPC servers via the new OPC XML-DA protocol.

TCP/IP
You can use TCP/IP worksheets to configure communication between your own project and other BOS projects. The TCP/IP Client and TCP/IP Server modules enable two or more projects to keep their databases synchronized using the TCP/IP protocol.

DDE
You can use DDE worksheets to communicate with other Microsoft Windows applications, such as Microsoft Excel, that support the Dynamic Data Exchange protocol.
Screen/Worksheet Editor

Use the powerful, object-oriented screen editor to create and edit a variety of screens and worksheets for your projects. You can input information using your mouse and keyboard, output control data to your processes, and automatically update screens based on data input from your processes.

Other screen editor features include:

- Simple point-and-click, drag-and-drop interface
- Grouping objects to preserve the construction steps of individual objects
- Editing objects without having to ungroup internal object components or groups
- Handling bitmap objects and background bitmaps
- Status line support in project windows and dialogs
About Tags and the Project Database

Tags are a core component of any BOS project. Simply put, tags are variables used by BOS to receive and store data obtained from communication with plant floor devices, from the results of calculations and functions, and from user input. In turn, tags can be used to display information on screens (and Web pages), to manipulate screen objects, and to control runtime tasks.

But tags are more than simple variables. BOS includes a real-time database manager that provides a number of sophisticated functions such as time-stamping of any value change, checking tag values against runtime minimum and maximum values, comparing tag values to alarming limits, and so on. A BOS tag has both a value and various properties that can be accessed, some at development and others only at runtime.

All tags are organized into one of the following categories, which are represented by folders on the Global tab of the Project Explorer:

- **Project Tags** are tags that you create during project development. Places where project tags are used include:
  - Screen tags
  - Tags that read from/write to field equipment
  - Control tags
  - Auxiliary tags used to perform mathematical calculations

- **Shared Database** tags are created in a PC-based control program and then imported into BOS’s tags database.
  For example you might create tags in SteepleChase and import them into BOS so BOS can read/write data from a SteepleChase PC-based control product.
  You cannot modify shared tags within BOS — you must modify the tags in the original PC-based control program, and then re-import them into the Tags database.

- **System Tags** are predefined tags with predetermined functions that are used for BOS supervisory tasks. For example,
  - Date tags hold the current date in string format
  - Time tags hold the current time in string format
 Most system tags are read-only, which means you cannot add, edit, or remove these tags from the database.
  To see a list of the system tags, select the Global tab in the Project Explorer, open the System Tags folder, and open the Tag List subfolder. The above figure shows a partial list of system tags.

After creating a tag, you can use it anywhere within the project, and you can use the same tag for more than one object or attribute.
Understanding the Tag Name Syntax

Observe the following guidelines when naming a tag:

- Your tag names **must be unique** — you cannot specify the same name for two different tags (or functions). If you type an existing tag name, BOS recognizes that the name exists and will not create the new tag.

- You must begin each tag name with a *letter*. Otherwise, you can use letters, numbers, and the underscore character (\_) in your tag name.

- You *cannot* use the following symbols in a tag name:

  `~ ! @ # $ % ^ & * ( ) - = \ + \ [ ] { } < > ?`

- You can use a maximum of 255 characters for a tag name or a class member name. You can use uppercase and lowercase characters. Tag names are *not* case sensitive. Because BOS does not differentiate between uppercase and lowercase characters, you can use both to make tag names more readable. (For example: `TankLevel` instead of `tanklevel`.)

- Tag names must be different from system tag names and math functions.

**Note:** Use the `@` character at the beginning of a tag name to indicate that the tag will be used as an *indirect tag* in the project.

Some valid tag examples include:

- `Temperature`
- `pressure1`
- `count`
- `x`
Choosing the Tag Data Type

Another consideration when designing a tag is what type of data the tag will receive. BOS recognizes the following, standard tag data types:

- **Boolean (one bit)**: Simple boolean with the possible values of 0 (false) and 1 (true). Equivalent to the "bool" data type in C++. Typically used for turning objects off and on or for closing and opening objects.

- **Integer (four bytes)**: Integer number (positive, negative, or zero) internally stored as a signed 32-bit. Equivalent to the "signed long int" data type in C++. Typically used for counting whole numbers or setting whole number values. Examples: 0, 5, −200.

- **Real (floating point, eight bytes)**: Real number that is stored internally as a signed 64-bit. Equivalent to the "double" data type in C++. Typically used for measurements or for decimal or fractional values.

- **String (alphanumeric data, up to 1024 characters)**: Character string up to 1024 characters that holds letters, numbers, or special characters. Supports both ASCII and UNICODE characters. Examples: Recipe product X123, 01/01/90, *** On ***.

You can also assign a new tag to a class that you have previously created.

You can find these tag types (and their respective icons) in the Global tab of the Project Explorer.
Using Array Tags

BOS tags can consist of a single value or an array of values.

**Note:** The maximum array size is 16384 as long as it does not exceed the maximum number of tags supported by the license (Product Type) selected for the project. Each array position (including the position 0) counts as one tag for licensing restrictions, because each position has an independent value.

An array tag is a set of tags with the same name, which is identified by indexes (a matrix of n lines and 1 column). The maximum array size depends on the product specification. You can use the following syntax to access an array tag:

```
ArrayTagName[ArrayIndex]
```

For example: `tank[0], tank[1], tank[2], and tank[500].`

**Note:** You must specify a maximum index for each array tag in the `size` column of any datasheet. You can specify n to indicate the array tag has positions from 0 to n. For example, if the size of TagA is 3, the tag elements could be `TagA[0], TagA[1], TagA[2], and TagA[3].`

Use the array tag whenever possible because it optimizes memory use and simplifies the configuration task. For example, if you want a display to monitor each tank, you could use array tags to configure a single display containing tags linked to any tank. For example (using the `tk` tag as an index containing the number of the tank): `pressure[tk], temperature[tk], and temperature[tk+1].`

An array index can be a tag, a numeric value, or an expression with the arithmetic operator "+".

**Note:** When you refer to an array with an index using the + arithmetic operation, you must use the following syntax:

```
ArrayTagName[NumValue1+NumValue2]
```

Where `NumValue1` and `NumValue2` can be an integer tag or a numerical constant. For example: `temperature[tk+2]` or `temperature[tk+6].`

Using array tags in any BOS task can save a significanct amount of project development time. For example, if you needed tag points related to the temperature of four tanks. The conventional configuration method is the following:

- `temperature1`: high temperature on tank 1
- `temperature2`: high temperature on tank 2
- `temperature3`: high temperature on tank 3
- `temperature4`: high temperature on tank 4

Using array tags simplifies this task, as follows:

- `temperature[j]`: high temperature on tank [j]

**Note:** When you create a four-position array tag, the system creates five positions (from 0 to 4). For example:

```
tag_example[15]  //start position=0, end position=15
```

Therefore, the `tag_example[15]` array has 16 elements.
When using another tag to reference the index of an array, if the value of the tag is outside the size of the array, then the following results are given:

- If \( \text{IndexTag} \) is greater than the size of the array, then \( \text{MyArray[IndexTag]} \) will point to the end position of the array; and
- If \( \text{IndexTag} \) is less than 0, then \( \text{MyArray[IndexTag]} \) will point to the start position of the array.

### Array Tags

An array tag consists of a set of tags that all have the same name, but use unique array indexes (a matrix of \( n \) lines and one column) to differentiate between each tag. An array index can be a fixed value, another tag or an expression. Maximum array sizes are determined by product specifications.

You can use array tags to:

- Simplify configurations
- Enable multiplexing in screens, recipes, and communication interfaces
- Save development time during tag declaration

You specify array tags in one of two formats:

- For a simple array tag, type: \( \text{ArrayTagName[ArrayIndex]} \)
- For a complex array tag (where the array index is an expression consisting of a tag and an arithmetic operation), type: \( \text{ArrayTagName[ArrayIndex+c]} \)

Where:

- \( \text{ArrayTagName} \) is the tag name;
- \( \text{[ArrayIndex]} \) is the unique index (fixed value or another tag);
- \( + \) is an arithmetic operation; and
- \( c \) is a numerical constant.

#### Note:

- You must specify a maximum index for each array tag by typing a value \( (n) \) in the Array Size column of an Project Tags datasheet or in the Array Size field on a New Tag dialog. (See "Creating project database Tags"). When you create an \( n \)-position array tag, BOS actually creates \( n+1 \) positions (from 0 to \( n \)). For example, if you specify \( \text{ArrayTag[15]} \), the array will have 16 elements, where 0 is the start position and 15 is the end position.
- You must not use spaces in an array tag.

When BOS reads a tag it begins with the first character and continues until it finds the first space or null character. Consequently, the system does not recognize any characters following the space as part of the array tag.

For example, if you type \( \text{a[second + 1]} \), BOS regards \( \text{a[second]} \) as the tag and considers it invalid because BOS does not find (recognize) the closing bracket. However, if you type \( \text{a[second} +1\text{]} \), this is a valid array tag.

You can specify an array tag wherever you would use a variable name. Also, because array tags greatly simplify configuration tasks and can save development time, we suggest using them whenever possible.

For example, suppose you want to monitor the temperature of four tanks. The conventional configuration method is:

- \( \text{temperature1} \) — high temperature on tank 1
• **temperature2** — high temperature on tank 2
• **temperature3** — high temperature on tank 3
• **temperature4** — high temperature on tank 4

You can use array tags to simplify this task as follows (where \([n]\) represents the tank number):
• **temperature\([n]\)** — high temperature on tank \([n]\)

The following table contains some additional examples of an array tag:

### Array Tag Examples

<table>
<thead>
<tr>
<th>Array Tag Example</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Tank[1].Tank[2].Tank[500]</td>
<td>Simple arrays, where the array indexes (1, 2, and 500) are numerical constants. For example, tank numbers.</td>
</tr>
<tr>
<td>Tank[tk]</td>
<td>A simple array, where the array index (tk) is a tag. For example, a tag representing the tank number.</td>
</tr>
<tr>
<td>Tank[tk+1]</td>
<td>A complex array, where the array index (tk+1) is an expression. For example, the value of tk (tank number) plus 1.</td>
</tr>
</tbody>
</table>

**Note:** When using another tag to reference the index of an array, if the value of the tag is outside the size of the array, then the following results are given:

- If **IndexTag** is greater than the size of the array, then **MyArray[IndexTag]** will point to the end position of the array; and
- If **IndexTag** is less than 0, then **MyArray[IndexTag]** will point to the start position of the array (i.e., **MyArray[0]**).
About indirect tags

BOS supports indirect access to tags in the database. For example, consider a tag $X$ of the String type. This tag can hold the name of any other tag in the database (that is, it can provide a pointer to any other type of tag, including a class type). The syntax for an indirect tag is straightforward: $\@\text{IndirectTagName}$. For example, assume that a tag named $X$ holds a "\text{TEMP}" string. Reading and/or writing to $\@X$ provides access to the value of the \text{TEMP} variable.

| Note: | Any tag created as a string-type tag is potentially an indirect tag (pointer). |

To refer to a class-type tag, you can declare a string-type tag that points to a class tag. For example:

<table>
<thead>
<tr>
<th>Class</th>
<th>TANK with members Level</th>
</tr>
</thead>
<tbody>
<tr>
<td>Tag</td>
<td>TK of the class TANK</td>
</tr>
<tr>
<td>Tag</td>
<td>XCLASS of the String type</td>
</tr>
</tbody>
</table>

To access the $\text{TK.Level}$ value, you must store the "$\text{TK.Level}$" value within the $\text{XCLASS}$ tag and use the syntax, $\@\text{XCLASS}$. You can also refer to a member of a class-type tag directly; identifying a class-type that points to a class member. For example:

<table>
<thead>
<tr>
<th>Class</th>
<th>TANK with members Level</th>
</tr>
</thead>
<tbody>
<tr>
<td>Tag</td>
<td>TK of the class TANK</td>
</tr>
<tr>
<td>Tag</td>
<td>XCLASS of the class TANK</td>
</tr>
</tbody>
</table>

To access the $\text{TK.Level}$ value, you must store the "$\text{TK}$" value within the $\text{XCLASS}$ tag and use the syntax, $\@\text{XCLASS.Level}$.

When creating tags for indirect use, place an $X$ in the tag column rather than creating them as strings. For the type, write the type of tag for which you are creating a reference. Follow the $\text{XCLASS}$ example: $\@Z\text{ Integer}, \@X\text{ Class:TANK}$.

Indirect Tags

Indirect tags "point" to other database tags (including class-type tags). Using indirect tags can save development time because they keep you from having to create duplicate tags (and the logic built into them).

You create an indirect tag from any string-type tag simply by typing the $\@$ symbol in front of the tag name $\@\text{TagName}$.

- To reference a simple tag, assume the $\text{strX}$ tag (a string tag) holds the value "\text{Tank}", which is the name of another tag, then reading from or writing to $\@\text{strX}$ provides access to the value of the \text{Tank} tag.

- To reference a class-type tag and member, you simply create a string tag that points to the class tag and the member. For example, if a tag $\text{strX}$ (a string tag) holds the value "\text{Tank.Level}", which is the name of the class tag, then reading from or writing to $\@\text{strX}$ provides access to the value of the \text{Tank.Level} member.

- You can also point directly to a class-type tag member; by identifying a class-type that points to a class member. For example: to access the $\text{Tank.Level}$ member of the class, you must store the "$\text{Tank}$" value within the $\text{strX}$ tag and use the syntax, $\@\text{strX.Level}$. 
Tutorial: Building a Simple Project

This section explains, using a step-by-step tutorial, how to build a simple project, as well as how to select and configure an I/O driver.
Creating a new project

This part of the tutorial shows how to create a new project, including how to give it a name and select the target platform.

1. Click the Application button in the top-left corner of the development environment, and then click New on the Application menu.
   The New dialog is displayed.

2. Click the Project tab.

3. In the Project name box, type the name of your project.
   For this tutorial, type Tutorial.
   The development application automatically creates a new directory of the same name and assigns your project file to that directory. (Notice the Configuration file text box in the figure.) To put your project file somewhere other than in the default projects folder, click Browse and navigate to the preferred location.

4. In the Product type list, select the type of project that you want to build.

5. Click OK.
   The New dialog is closed and the Project Wizard dialog is displayed.

6. In the Template list, select Empty Application.

7. In the Resolution list, select 640 x 480.

8. Click OK.
   The Project Wizard dialog is closed and the new project is created in the development environment.
Specifying the startup screen

This part of the tutorial shows how to open the project settings and then specify which screen should be displayed on startup.

- Use the **Information** tab to provide information that identifies the project (such as project description, revision number, Company name, Author's name, field equipment, and general notes).
- Use the **Options** tab to specify generic settings for the project, such as the Target System, Automatic Translation, Alarm history and Events, Default Database and Shared Tags.
- Use the **Viewer** tab to enable/disable the runtime desktop parameters.
- Use the **Communication** tab to specify communication parameters relating to the project in general.
- Use the **Preferences** tab to enable/disable warning messages when using the development application.

To specify the startup screen:

1. On the **Project** tab of the ribbon, in the **Settings** group, click **Viewer**.
   The **Project Settings** dialog is displayed with the **Viewer** tab selected.

2. In the **Startup screen** box, type **main.scc**.
   When you run the project, it will automatically display the main screen (or whichever screen you specify) first. You can specify a screen before you create it, but if the screen has been created, then you can also select it from the list.

3. Click **OK**.
Creating tags

This part of the tutorial shows how to create new tags by adding them to the Project Tags datasheet. A tag is any variable that holds a value. All tags created in a project are stored in the Project Tags folder, on the Global tab of the Project Explorer.

1. In the Project Explorer, click the Global tab.
2. Double-click Project Tags to expand the folder.
3. Double-click Datasheet View to open the Project Tags datasheet.
4. Use the following parameters to create a tag for the sample project.
   a) Name: Specify a unique tag name. For this tutorial, type Level.
   b) Array: Specify the top array index of the tag. (Simple tags have an Array of 0.) For this tutorial, type 3.
      Each array index relates to one of the three tanks:
      • Level[1] is the level of Tank #1
      • Level[2] is the level of Tank #2
      • Level[3] is the level of Tank #3
      You will not use Level[0] in this tutorial, even though it is a valid tag.
   c) Type: Specify the data type of the tag: Boolean, Integer, Real, String, or Class. For this tutorial, select Integer.
   d) Description (optional): Type a description of the tag for documentation purposes only.
   e) Scope: Specify how the tag is managed between the Server and the Thin Client stations.
      • Select Local if you want the tag to have independent values on the Server and Client stations.
      • Select Server if you want the tag to share the same value on the Server and Client stations.
      For this tutorial, select Server.

5. Save and close the Project Tags datasheet.

You will create additional tags as you build the project.

Tip: You can sort the data in the Project Tags datasheet or insert/remove additional columns by right-clicking on it and then choosing the applicable option from the pop-up menu.
Creating the main screen

This part of the tutorial shows how to create your first screen, which will contain a single button that opens another screen.

1. In the Project Explorer, click the Graphics tab.

2. Right-click Screens, and then click Insert on the shortcut menu. The development application stores all screens created for a project in this Screens folder. The Screen Attributes dialog is displayed.

3. Use this dialog to set screen properties such as size and type. For this tutorial, click OK to accept the default settings. The Screen Attributes dialog is closed, and the new screen is opened in the workspace for editing.

4. On the Graphics tab of the ribbon, in the Screen group, click Background Color. A standard color picker is displayed.
5. In the color picker, select a light gray color.

That color is applied to the screen.

**Drawing the main screen’s title**

This part of the tutorial shows how to draw the main screen’s title using a Text object.

1. On the **Graphics** tab of the ribbon, in the **Active Objects** group, click **Text**. Your mouse cursor changes from an arrow to a crosshair.

2. Click on the screen, type **Welcome to the Tutorial Application**, and then press Return. This creates a new Text object with the specified text.

3. Double-click the object to open its **Object Properties** dialog.

   - Double-clicking on any screen object opens an **Object Properties** dialog containing the properties for that object. The properties shown in the dialog change depending on the type of object.
   - The **Object Properties** dialog also contains a pin button that controls whether this dialog remains open. The button changes state (and function) each time you click on it, as follows:
     - When the pin button is released, the focus is passed to the object on the screen as soon as it is selected. It is recommended that this button is kept released when you want to manipulate the objects (Copy, Paste, Cut, or Delete). Although the **Object Properties** dialog is on the top, the keyboard commands (Ctrl+C, Ctrl+V, Ctrl+X, or Del) are sent directly to the objects.
     - When the pin button is pressed, the focus is kept on the **Object Properties** dialog, even when you click the objects on the screen. We recommend you keep this button pressed when you want to modify the settings of the objects. You can click an object and type the new property value directly in the **Object Properties** dialog (it is not necessary to click on the window to bring focus to it). Also, when the pin button is pressed, the **Object Properties** dialog does not automatically close when you click on the screen.

4. Click **Fonts** to open **Font** dialog, and then specify the font settings.
   For this tutorial...
   - Font is **Arial**
   - Font style is **Regular**
   - Size is **20**
• Color is **Blue**

![Specifying the font settings](image1)

5. Click **OK** to close the *Font* dialog. 
The font settings are applied to the Text object.

![Font settings applied to Text object](image2)

6. Close the *Object Properties* dialog (i.e., click the Close button in the dialog box's top-right corner).

**Drawing a button to open another screen**

This part of the tutorial shows how to draw and configure a button that will open another screen.

1. On the **Graphics** tab of the ribbon, in the **Active Objects** group, click **Button**.
   Your mouse cursor changes from an arrow to a crosshair.

2. Click and hold on the screen, and then drag the cursor to draw the Button object.

3. Double-click the object to open its *Object Properties* dialog.
4. In the Caption box, type the following text: Click here to open the synoptic screen.

![Adding a caption to the button](image)

5. Click Command. The Object Properties dialog changes to show the properties for the Command animation.

6. In the Type list, select Open Screen.

7. In the Open Screen box, type synoptic.scc.

![Configuring an Open Screen command on the button](image)

You can specify a screen that you have not yet created.


**Saving and closing the main screen**

This part of the tutorial shows how to properly save and close a screen.

1. Click the Application button at the top-left of the development environment, and then click Save on the Application menu. A standard Windows Save dialog is displayed.

2. In the File name box, type main.

3. Click Save. The file is saved in your project folder (at `<project name>\Screen\main.scc`), and the Save dialog is closed.

4. Click the Application button at the top-left of the development environment, and then click Close on the Application menu.
Creating the synoptic screen

This part of the tutorial show how to create your second screen, which will include an animated tank of liquid and some basic controls for that tank.

1. In the **Graphics** tab of the **Project Explorer**, right-click the **Screens** folder, and then click **Insert** on the shortcut menu.
   The **Screen Attributes** dialog is displayed.

2. Use this dialog to set attributes such as size and type.
   For this tutorial, click **OK** to accept the default settings.

3. Click the Application button at the top-left of the development environment, and then click **Save As** on the Application menu.
   A standard Windows **Save As** dialog is displayed.

4. In the **File name** box, type **synoptic**.

5. Click **Save**.
   The file is saved in your project folder (at `<project name:gt;\Screen\synoptic.scc`), and the **Save** dialog is closed.

Do not close the screen like you did the main screen when you saved it. You still need to draw the synoptic screen.

**Drawing the synoptic screen's title**

As in a previous part, this part of the tutorial shows how to draw the synoptic screen's title using a Text object.

1. On the **Graphics** tab of the ribbon, in the **Active Objects** group, click **Text**.

2. Click on the screen, type **Synoptic Screen**, and then press Return.

3. Double-click the object to open its **Object Properties** dialog.

4. Click **Fonts** to open **Font** dialog, and then specify the font settings.
   For this tutorial...
   
   • Font is **Arial**
   • Font style is **Bold**
   • Size is **20**
   • Color is **Blue**

5. Click **OK** to save the font settings and close the dialog.

6. Close the **Object Properties** dialog.

7. Move the Text object to the top left corner of the screen.

8. Click the Application button at the top-left of the development environment, and then click **Save** on the Application menu.

This figure shows how your screen should look after you have drawn the screen title.

![Finished screen title](image)

**Drawing "Date" and "Time" displays**

This part of the tutorial shows how to draw "Date" and "Time" displays by linking Text objects to system tags. **Date** and **Time** are system tags that hold the current date and time of the local station. These tags are available to any project.

1. On the **Graphics** tab of the ribbon, in the **Active Objects** group, click **Text**.
2. Click on the screen, type Date: ##########, and then press Return.

3. Double-click the object to open its Object Properties dialog.

4. Click Text Data Link.
   The Object Properties dialog changes to show the properties for the Text Data Link animation.

5. In the Tag/Expression box, type Date.

   ![Object Properties dialog]

   **Specifying the Date system tag**

   During runtime, the project replaces the ########## characters of the Text object with the value of the system tag Date.


7. On the Graphics tab of the ribbon, in the Active Objects group, click Text.

8. Click on the screen, type Time: ##########, and then press Return.

9. Double-click the object to open its Object Properties dialog.

10. Click Text Data Link.
   The Object Properties dialog changes to show the properties for the Text Data Link animation.

11. In the Tag/Expression box, type Time.

   ![Object Properties dialog]

   **Specifying the Time system tag**

   During runtime, the project replaces the ########## characters of the Text object with the value of the system tag Time.


13. Click the Application button at the top-left of the development environment, and then click Save on the Application menu.

   This figure shows how your screen should look after you have created the date and time objects.

   ![Finished date and time objects]
**Placing an "Exit" icon**

This part of the tutorial shows how to place an icon (by selecting and configuring a Linked Symbol) that allows the user to exit the project.

1. On the **Graphics** tab of the ribbon, in the **Libraries** group, click **Symbols**.
   The symbols library is displayed.
2. In the Symbols menu tree, open the **System Symbols** folder and then open the **Icons** sub-folder.
3. In the Icons sub-folder, select **exit01**.
   The symbol will be displayed in the symbol viewer to the right of the menu tree.

4. Click on the symbol.
   The mouse cursor will change to show that the symbol is ready to be placed in a screen.
5. Switch back to the screen where you want to place the symbol and then click in it.
   The symbol is placed as a Linked Symbol object.

6. With the object still selected, click **Command** (on the **Graphics** tab of the ribbon, in the **Animations** group) to apply this animation to the object.
7. Double-click the object to open its **Object Properties** dialog.
8. In the **Type** list, select **VBScript**.
9. In the **On Down** box, type `$Shutdown()`.
   *Shutdown* is one of BLUE Open Studio’s built-in scripting functions, but it can be used within VBScript by prefacing it with a dollar sign ($).

10. Close the **Object Properties** dialog.
11. Click the Application button at the top-left of the development environment, and then click **Save** on the Application menu.

Now, when a user clicks this icon during runtime, the project will stop and exit to the station’s desktop.
**Testing the project**

This part of the tutorial shows how to test the project so far.

1. Click the Application button at the top-left of the development environment, and then click **Close > Close All** on the Application menu.
   
   All open worksheets are closed.

2. On the **Home** tab of the ribbon, in the **Local Management** group, click **Run**.
   
   The project runs and the startup screen is displayed.

3. Click the button to open the synoptic screen.
   
   The synoptic screen is displayed.

4. Click the exit icon to shut down the project.

If any part of the project does not work as expected, switch back to the development application (**ALT+TAB**) and then click **Stop** on the Home tab of the ribbon.

**Placing an animated tank**

This part of the tutorial shows how to select an animated tank from the Symbol Library and place it on the screen (similar to how you selected and placed the "Exit" icon), then associate some project tags with the tank’s properties.

1. In the **Graphics** tab of the **Project Explorer**, expand the **Screens** folder.

2. Double-click **synoptic.scc**.
   
   The synoptic screen worksheet is reopened for editing.

3. On the **Graphics** tab of the ribbon, in the **Libraries** group, click **Symbols**.

4. In the Symbols menu tree, open the **System Symbols** folder and then open the **Tanks** sub-folder.

5. Browse the tank symbols and choose one.
   
   You may choose any tank symbol that you like; they all function basically the same.

6. Click the symbol.
   
   The mouse cursor will change to show that the symbol is ready to be placed in a screen.

7. Switch back to the screen where you want to place the symbol and click in it.
   
   The symbol is placed as a Linked Symbol object.
8. Double-click the object to open its *Object Properties* dialog.

8. Double-click the object to open its *Object Properties* dialog.

![Object Properties dialog](image)

*The tank symbol's properties*

A tank is an arrangement of different objects and animations (for example a rectangle, a bar graph, etc.), all combined together as a Linked Symbol. You can modify the properties of this symbol by editing the properties list. For this tutorial, you will modify the tag associated with the tank level.

9. For the property *TagLevel*, delete the existing value and then type *Level[Index]*.

Note that you do not need to reopen the Project Tags datasheet to create tags as you develop the project. Because you have not previously created the tag *Index* in the Project Tags database, an alert message asks you if you would like to create it.

10. Click *Yes*.

A *New Tag* dialog is displayed.

11. Configure the new tag with *Array* as 0, *Type* as *Integer*, and *Scope* as *Local*.

![New Tag dialog](image)

*Configuring a new tag*

12. Click *OK* to close the *New Tag* dialog.

You can use the tag *Index* to set the array position of the tag *Level*, and show the level for any of the three tanks in the same object:

- When *Index* equals 1, the tank object shows the level of Tank #1 (i.e., *Level[1]*);
- When *Index* equals 2, the tank object shows the level of Tank #2 (i.e., *Level[2]*); and
- When *Index* equals 3, the tank object shows the level of Tank #3 (i.e., *Level[3]*).

Also, because the tag scope is local, the tag can have different values for the Server and Client stations at the same time. Consequently, the local user (i.e., the Server station) can be monitoring the level of Tank #1 while the remote user (i.e., the Client station) is monitoring the level of Tank #2.

13. Close the *Object Properties* dialog.

14. Click the Application button at the top-left of the development environment, and then click *Save* on the Application menu.
This figure shows how your screen should look after you’ve created the tank object.

![Finished tank object](image1)

**Placing a level slider**

This part of the tutorial shows how to select a slider control from the Symbol Library and then connect it to the animated tank.

1. On the **Graphics** tab of the ribbon, in the **Libraries** group, click **Symbols**.
2. In the Symbols menu tree, open the **System Symbols** folder and then open the **Sliders** sub-folder.

![Selecting a slider symbol](image2)

3. In the Sliders sub-folder, select a slider control. You may select any slider you like; they all function basically the same way.
4. Click on the symbol. The mouse cursor will change to show that the symbol is ready to be placed in a screen.
5. Switch back to the screen where you want to place the symbol and click in it. The symbol is placed as a Linked Symbol object.
6. Double-click the object to open its **Object Properties** dialog.
7. For the property **TagName**, delete the existing value and then type `Level[Index]`. Just as with the tank, you need to modify the symbol property associated with the slider level.
8. Close the **Object Properties** dialog.
9. Click the Application button at the top-left of the development environment, and then click **Save** on the Application menu.

This figure shows how your screen should look after you’ve created the level slider object.

![Finished level slider object](image3)
**Drawing a tank selector**

This part of the tutorial shows how to draw a text input box that can be used to change which real-world tank is represented by the animated tank on the screen.

1. On the **Graphics** tab of the ribbon, in the **Active Objects** group, click **Text**.
2. Click on the screen, type **Tank: #**, and then press Return.
3. Double-click the object to open its **Object Properties** dialog.
4. Click **Text Data Link**.
   The **Object Properties** dialog changes to show the properties for the Text Data Link animation.
5. In the **Tag/Expression** box, type Index.
6. Select the **Input Enabled** option.
   This allows the operator to enter a new value for the tag during runtime.
7. In the **Minimum Value** box, type 1.
8. In the **Maximum Value** box, type 3.

![Configuring the "Tank" text input](image)

9. Close the **Object Properties** dialog.
10. Click the Application button at the top-left of the development environment, and then click **Save** on the Application menu.

   This figure shows how your screen should look after you’ve created the tank selector object.

![Finished tank selector object during runtime](image)

**Testing the project**

This part of the tutorial shows how to test the project again with the animated tank, the level slider, and the tank selector.

1. Click the Application button at the top-left of the development environment, and then click **Close > Close All** on the Application menu.
   All open worksheets are closed.
2. On the **Home** tab of the ribbon, in the **Local Management** group, click **Run**.
   The project runs and the startup screen is displayed.
3. Click the button to open the synoptic screen.
   The synoptic screen is displayed.
4. Type the tank number (1, 2, or 3) in the Tank label, and then use the slider to adjust the tank level. Note that you can view/adjust the level of each tank independently.

5. Click the exit icon to shut down the project.

If any part of the project does not work as expected, switch back to the development application (ALT+TAB) and then click Stop on the Home tab of the ribbon.
Configuring the communication driver

This part of the tutorial shows how to select and configure a driver to communicate with an external I/O device.

1. In the Project Explorer, click the Comm tab.
2. Right-click the Drivers folder, and click Add/Remove Drivers on the shortcut menu. The Communication Drivers dialog is displayed.
3. Select a driver from the Available drivers list, and then click Select. For this tutorial, select MODBU. The driver is moved to the Selected drivers list.

4. Click OK. The Communication Drivers dialog is closed, and the driver is added to the Drivers folder in the Project Explorer.
5. In the Project Explorer, right-click the MODBU folder, and then click Settings on the shortcut menu.

The Communication Settings dialog is displayed.
6. Configure the communication settings as needed for the target device. For this tutorial, accept the default settings.

   - **Note:** For more information about a specific driver, click **Communication Drivers** on the **Help** tab of the ribbon.

7. Click **OK** to close the dialog.

8. In the Project Explorer, right-click the **MODBU** folder and then click **Insert** on the shortcut menu. A new driver worksheet named **MODBU001.drv** is created and opened for editing.

9. Configure the worksheet header:
   a) In the **Description** box, type **Tutorial Modbus**. This setting is for documentation only; it does not affect the runtime project in any way.
   b) In the **Enable Read When Idle** box, type **1**. This setting is a trigger that takes a Boolean value. A value of 1 — either entered manually as above or evaluated from a tag/expression — forces your project to continue reading tag values from the target device even when there are no changes in value.
   c) In the **Enable Write On Tag Change** box, type **1**. This setting is also a trigger. A value of 1 forces your project to write tag values to the target device only when those values change, rather than continuously. This saves system resources and improves performance during runtime.
   d) In the **Station** box, type **1**. This indicates the I/O device number to be accessed by this driver. Typically, the PLC is specified as Device #1.
   e) In the **Header** box, type **4X:0**. You must use a driver-specific format. The format for the MODBU driver is:

   \[
   \text{register\_type}:\text{initial\_offset}
   \]

<table>
<thead>
<tr>
<th>Register Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>0X</td>
<td>Coil Status</td>
</tr>
<tr>
<td>1X</td>
<td>Input Status</td>
</tr>
<tr>
<td>3X</td>
<td>Input Register</td>
</tr>
<tr>
<td>4X</td>
<td>Holding Register</td>
</tr>
<tr>
<td>ID</td>
<td>Slave ID Number</td>
</tr>
</tbody>
</table>

10. In the worksheet body, enter the tags and their associated device addresses — for each tag:
   a) In the **Tag Name** field, type the name of the project tag.
   b) In the **Address** field, type the value to be added to the header to form the complete device address.
11. Click the Application button at the top-left of the development environment, and then click **Save** on the Application menu.

12. When prompted to choose the driver sheet number, type **1** and then click **OK**.

### Monitoring device I/O during runtime

This part of the tutorial shows how to monitor device I/O during runtime by using the Log window.

1. On the **Home** tab of the ribbon, in the **Local Management** group, click **Run**. The project runs and the startup screen is displayed.

2. Press **ALT+TAB** to switch back to the development application.

3. Right-click in the **Output** window, and then click **Settings**. The Log Settings dialog is displayed.

4. Select the **Field Read Commands**, **Field Write Commands**, and **Protocol Analyzer** options.

5. Click **OK** to close the Log Settings dialog.

You can now monitor the device I/O during runtime.

---

<table>
<thead>
<tr>
<th>Tag Name</th>
<th>Address</th>
<th>Complete Device Address</th>
</tr>
</thead>
<tbody>
<tr>
<td>Level[1]</td>
<td>1</td>
<td>4X.1 (Holding Register 1)</td>
</tr>
<tr>
<td>Level[2]</td>
<td>2</td>
<td>4X.2 (Holding Register 2)</td>
</tr>
<tr>
<td>Level[3]</td>
<td>3</td>
<td>4X.3 (Holding Register 3)</td>
</tr>
</tbody>
</table>