1 General

1.1 Scope

The purpose of this Guide Form Specification is to outline some of the requirements and considerations in the selection and implementation of a monitoring and control system. A monitoring and control system provides the infrastructure required to support the collection and sharing of data and information from the various devices to the enterprise level. The technology base that the software system is built upon is as equally important as the functionality delivered by the system. A sound technology base built upon industry accepted standards help assure a smooth transition and migration path as new technologies evolve. Since 1997, Microsoft and its technologies has become a strong player on the factory floor and offers the greatest promise of a flexible future migration path.

A monitoring and control system makes it possible to exchange data in the form of discrete and analog values from I/O devices and/or control devices such as IED (Intelligent Electronic Device) add programmable logic controllers. The system also allows information to be easily shared between computer systems and applications through an open systems design and architecture. The system also makes it possible to support modular applications such as graphical display, alarming, logical functions, analysis, data handling operations and can communicate with enterprise or external systems over a network. Sharing of data should be capable through Web based browsers as well as standard Client Viewers.

1.2 Definitions

- HMI: Human Machine Interface. Used to provide a graphic representation of data from a process and to accept user commands to be fed back to the process.
- Ethernet: A very high performance local area network standard providing the two lower levels of the ISO/OSI seven layer reference model, the physical layer and the data link layer.
- TCP/IP: a protocol widely used across Ethernet networks for connecting computers and programmable controllers.
- Web Browser: A client application that provides a user interface via the World Wide Web. Google Chrome and Microsoft Internet Explorer are two popular examples.
- Thin Client: In client/server applications, a client designed to be especially small so that the bulk of the data processing occurs on the server. The term thin client refers to a client designed to be as small as possible.
- Data Concentrator: A physical device that translates analog and digital information from attached I/O devices to a protocol that can be used with an HMI.
- Communications Protocol: A formal set of conventions governing the control of Inputs and Outputs between the two communicating processes.
- Network: An interconnected group of nodes, a series of devices, nodes or stations connected by communications channels.
- Operating System: A program that controls the entire overall operation of the computer system hardware/software.

2 Industry Standards

Industry standards lay the cornerstone for the technology base of a monitoring and control software system. The system should conform to and take advantage of industry standard and de-facto standards. These should include, but not be limited to:

- ODBC
- ActiveX
- HTML5
- COM/DCOM
- DDE and AdvanceDDE
- C, C++, C# programming language
- Visual Basic®
- Microsoft Windows
- Microsoft Windows IOT
- TCP/IP
- OPC DA
- OPC UA
- OPC Alarm and Events (A&E)
- ISA 18.2
- ISA 101
- XML
- .NET
- IEC 61850
- DNP 3.0
- IEC-62443-2-4
- ISA 101

3 System Architecture

3.1 Overall Design

The system should be designed to pass data and information between a variety of computer platforms and operating systems. It should support industry standards, be modular in design, and provide application program interfaces to allow easy customization.

3.2 Client/Server

In order to reduce the overhead of multiple independent systems, the software system should support a distributed Client / Server architecture. This should include the support of Human Machine Interfaces, Data Servers, Viewers (clients), and Web based Thin Clients. Each of these components should work in conjunction with the others using industry standard Ethernet networks.

3.3 HMI Client/Server

The monitoring and control software system should a **true** client server HMI architecture. HMI Servers should poll and collect data from devices. This data is to be seamlessly shared among other HMI Servers and to Viewers without requiring the duplication of data values in another node. HMI Viewers should act as clients to the HMI Servers, seamlessly receiving their data.

In a true client server environment, data is to be configured once. The HMI should support one time configuration of data points. For example, points to be polled from a device should be configured once on that server and other Servers or Viewers need only reference that data item to use it for their applications.

Alarms should be configured once and served across multiple HMI Servers and Viewers.

HMI Viewers should be able to simultaneously access data from one or more HMI Servers. Data from multiple Servers may be used within the same graphic screen. Viewers should require only a valid TCP/IP connection to the Server to obtain this data. This TCP/IP link may typically be across a local area Ethernet network (LAN) although it may be across a wide area network (WAN) and make use of telephone lines, radio, and/or satellite links.

3.4 Human Machine Interface Servers

The HMI Server software should be configurable to provide for the monitoring and control of all points, loops, and systems through graphic display screens and hard copy (printer output) reports. These should include but not be limited to:

- Parameter Displays for signal control
- Control Loop Status Displays
- Real Time and Historical Data Trend Displays
- Event Displays and Log Reports
- Alarm Displays and Log Reports
- Equipment Diagnostic Displays and Reports

HMI Servers should be capable of operating independently but should be configurable to share data with other HMI Servers or Enterprise Server systems.

3.5 Enterprise Servers

The software system should support the use of Enterprise Servers. Enterprise Servers should be capable of consolidating data from HMI Servers without the need for reconfiguring point values. Servers should provide a broad view of information within a facility and should be capable of supporting host or database applications such as Historian SPC and Production Tracking. In addition, Enterprise Servers should be capable of supplying data services to networked viewers.

3.6 Viewers

Viewer Systems should provide occasional system users with easy access to plant floor data using the same graphical user interface as Enterprise and HMI Servers.

Viewers should be capable of both displaying and modifying data.

Viewers should have the ability to access OPC data directly without accessing the SCADA server.

3.7 Web Based Thin Client Viewers

Web based Thin Client Viewers should provide the capability of viewing the graphic screens from the software system. The user should not need to recreate screens for the Thin Clients. The software system must be able to support the display of the graphic screens without the need for the user to recreate the screens in html documents.

The web based clients must support the full capability of the thick client and offer the ability to have security access to the capability of the thick client or specifically be designed for a modernized HTML5 environment.

The Thin Client server capability must be able to be deployed in a redundant manner using Windows Clustering to remove the single point of failure.

The Thin client server must allow more than 100 concurrent clients to be connected to the SCADA server.

3.8 Computer Platforms

The computer platform for HMI servers should be a PC compatible personal computer running Windows operating systems. Runtime Viewers should also be supported on Windows operating systems.

3.9 User Management

In supporting a client / server architecture, the product must be able to handle multiple users with different log ons and security. The system shall also be able to log a person off after a configured period of time without interaction with the system. The system shall also allow configuration to use users in the Windows Domain, and authenticate against that domain.

3.10 Secure Architecture

A secure reference architecture and guidelines shall be provided by the vendor to provide optimal security architecture and configuration. The product must also be Achilles certified for Achilles Practices Certification to meet IEC-62443-2-4 standards.

4 Technology

4.1 Microsoft Technologies

Technology is the foundation of a software system and critical to the current as well as future functionality it is capable of delivering. Choosing a system that is based on industry driven standards helps assures compatibility with existing systems as well as provides a roadmap for the future.

Microsoft has been a key player in driving the technologies used in systems on the facility floor.

Microsoft has either driven or helped sponsored the following open system standards:

- ODBC
- DDE
- OLE
- ActiveX
- COM/DCOM
- OPC
- Windows Metafile
- XML
- .NET
- WPF

ODBC - Open Database Connectivity is one of the fundamental standards that empower a software system. ODBC provides the means for applications to easily access and retrieve information from relational databases. Data that is collected by a software system is transformed into information that users can use. This information must be stored and retrieved from a database. ODBC provides an open systems approach to the storage and retrieval of your system information.

DDE, OLE, and ActiveX. DDE, Dynamic Data Exchange, is a standard that was developed to allow applications to pass data in a real time fashion. This then evolved to OLE - Object Linking and Embedding. OLE allows for one application to be contained within another. A typical example of OLE is the ability to embed a Microsoft Excel chart in a Word document. Once embedded, it is easy to access the other application. In the case of the Excel chart, updates to it can be automatically reflected in the Word document. ActiveX takes the evolution one step further by providing the ability to embed a self-contained application (object) within another application. ActiveX objects are often user interface applications that receive and send data and information from the software system while allowing the user a view and interact with the information. It is therefore critical that the software system support DDE, OLE, and ActiveX as these standards provide an open systems approach allowing the user to select the best components and object to integrate the application.

OPC is a standard that has gained acceptance among vendors of HMI and process automation products. OPC builds on the success and strength of the Microsoft operating systems and OLE (Object Linking and Embedding) and other operating systems with the introduction of OPC UA. The design of OPC was initiated in 1995, and was targeted at providing a standard for interoperability between control applications, field systems and devices, and business and office applications. OPC is endorsed by Microsoft and the OPC Foundation which developed the technical specification. The OPC Foundation now includes over 450 members with broad representation from the process control industry.

The sign of a true open systems based monitoring and control system is its capabilities of acting as an OPC Client as well as an OPC Server. As an OPC Client, the system must be able to collect data from third party OPC data collection servers. As an OPC Server, the software system must be able to share the data and information it manages with other third party programs.

OPC Alarm and Events (A&E) provides alarm and event notifications on demand (in contrast to the continuous data flow of Data Access). These include process alarms, operator actions, informational messages, and tracking/auditing messages. That system shall be able to be act as an OPC A&E server or OPC A&E client.

OPC UA is the next evolution of OPC and encompasses the Data, alarm and events, historical data access and object model into a single standard. This standard also includes secure certificate based communications The system shall be able to act as an OPC UA Server and OPC UA client and provide the Global Discovery Capability required for certificate authority and exchange.

COM (Component Object Model) is a Microsoft software architecture that allows applications to be built from software components. COM is the foundation behind higher-level software services, like those provided by OLE. From its original application on a single machine, COM has expanded to allow access to components on other systems. Distributed COM (DCOM), introduced in 1996, makes it possible to create networked applications built from components. DCOM is used today in applications ranging from cutting edge medical technology to traditional accounting and human resources systems. COM and DCOM provides the architecture that enables true client/server based software systems.

Windows Metafiles is a standard for vector-based graphics. Metafiles are represented as collections of lines rather than pixels, so you can manipulate them without the distortions common to bitmap (raster) graphics. Software systems that support the Metafile graphic format are capable of importing graphic objects from other packages such as AutoCAD, Visio, and PowerPoint. The support for these third party graphics can greatly reduce the time to commission a system by allowing the reuse of existing graphical design work.

XML is a standard that allows software developers to easily describe and deliver rich, structured data from any application in a standard, consistent way. XML does not replace HTML; rather, it is a complementary format. XML enables a new generation of Web-based data viewing and manipulation applications. The HMI product should provide XML server capabilities to provide data collected from the shop floor out to other applications using the XML format. In addition the HMI should provide the ability to view point and alarm reports in XML format in standard web browsers.

4.2 .NET

.NET, commonly referred as the .NET Framework, is a software framework that runs primarily on Microsoft Windows. It includes a large library and supports several programming languages which allows language interoperability (each language can use code written in other languages). Programs written for the .NET Framework execute in a software environment (as contrasted to hardware environment), known as the Common Language Runtime (CLR), an application virtual machine that provides important services such as security, memory management, and exception handling. The class library and the CLR together constitute the .NET Framework. The .NET Framework's Base Class Library provides user interface, data access, database connectivity, cryptography, web application development, numeric algorithms, and network communications. Programmers produce software by combining their own source code with the .NET Framework and other libraries. Object Technologies

ActiveX is a standard that defines self-registering, in-process COM components that often have a visual element either at design time or run time. It is important that the system supports ActiveX and provides an ActiveX container application. This allows native or third party ActiveX controls the ability to communicate with the software system. The use and support of ActiveX controls can greatly expand the functionality and applications the software system supports.

In addition to supporting ActiveX, the graphical interface for the system must be object oriented. It must be possible for the user to construct their own objects and incorporate point information, animation, and screen based scripts. It must be possible to create master objects that then can be linked into other screens. Any changes to the master object shall be reflected in all other screens the next time the screen is accessed. It must be possible to create and configure object classes that outline a data structure consisting of both static attributes

and variable information. For example, a user should be able to create an object for a motor that identifies attributes for information such as the manufacturer and horsepower rating as well as hold variable data for the actual speed and status of the motor. Once an object is created, the user must be able to freely instantiate several instances of the object.

4.3 Web Technologies

Internet and Web technologies have changed the way business is being conducted. It is important for the software system to have the capability of sharing and transmitting its collected and maintained information to internet users. It must be possible for the information of the software system to be shared in standard Mozilla or Internet Explorer Web browsers without the need to load any additional client software on the user's PCs. Users must be able to view the information as well as with sufficient password protection, enter or modify information as well.

4.4 Wireless Technologies

The software system should support a wireless Ethernet architecture. This will allow mobile users access to the system's information on laptops or Windows based hand-held PCs, tablets or mobile phones. Wireless Ethernet technologies supported should be spread spectrum frequency hopping based rather than spread spectrum direct sequence to avoid possible interference and performance problems on the manufacturing floor.

4.5 Programming and Interfaces

An open systems approach to the design and implementation of the software system is critical. The system must not only adopt open system standards such as ODBC, OLE, OPC, ActiveX, .NET and COM/DCOM, but must also provide open APIs (Application Program Interfaces). APIs shall allow users to develop their own direct interfaces with the data and alarm information collected, managed, and maintained by the software system. Programming calls used in the APIs shall be the same as those used internally in the construction of the software system code. This common design shall permit a seamless integration of custom or third party applications with the software system.

5 Real Time Data Management

A key capability of a monitoring and control software system is the ability to provide a real time, distributed, memory resident database of current process data values. These data ("point" or "tag") values may be from definable device points representing the value of a physical data collection item on a resource, or virtual points representing values calculated from one or more device point values used in a mathematical expression.

5.1 Data Sharing

Point values should be stored, retrieved and manipulated across one or more computers using the software system's distributed architecture. Data integrity must be automatically and continuously ensured.

Writing of custom software should not be required to provide simple data sharing among HMI Servers, Enterprise Servers and or Viewers. Viewers should simply "connect" to Servers to obtain data. Enterprise Servers should have configurable support for collecting data from HMI Servers.

The software system must provide Application Program Interfaces that are published and available to permit users to write custom software to support interfacing the HMI and Enterprise Servers to other computer systems and applications. To ensure that these "API's" provide adequate support for device data collection, it is recommended that the software system and HMI are the same vendor.

The HMI should also support data sharing where HMI acts as the "server" and applications such as Microsoft Excel act as the "client", or HMI can be a client to those application.

5.2 Data Collection Methods

The software system needs to support a wide range of data collection methods. Data collection methods should include scheduled polling, on-change, unsolicited, timed interval, “on demand”, triggered reads, and array support, among others. Engineering Unit conversions on collected data and reverse engineering units conversions for setpoints are required.

HMI Servers should have configurable support for obtaining data from supported plant floor devices.

The HMI should also support data collection from DDE servers using both the DDE and AdvancedDDE protocol standards. In this mode, the HMI operates as a client. This capability permits the use of third party software to extended data collection support.

The HMI should be capable of acting as an OPC Client as well as an OPC Server. As an OPC Client it should be capable of collecting data from third party OPC servers. As an OPC Server it should be capable of serving the information it is maintaining out to other third party software packages acting as OPC Clients. The OPC can be OPC UA, OPC DA or OPC AE.

5.3 Required Data Types

The software system should support the following Data Types:

- Global
- Floating point
- Analog (Signed and Unsigned)
- Discrete
- String
- Arrays
- Structures

5.4 Device Communications

The software system must be capable of supporting hundreds of different models, makes, manufacturers, and protocols of programmable and industrial control devices. Support for the following should be included as a minimum:

- Modbus
- GE
- Emerson
- Allen-Bradley
- Modicon
- Siemens
- DNP (for Water or Power Applications)
- IEC 61850 (for Power Applications)

To permit integration of any device the system shall support open architecture standards and application programming interfaces to allow the development of custom drivers as required.

As described earlier, the software system should also support use OPC servers to provide broader device communications.

5.4.1 System Points

The system should provide a wide range of pre-defined system points that include a wide variety of topics including alarms, date and time, project, and computer information. System points should be available to be used by the standard HMI components such as graphics and scripting. The software system should accommodate for system points in a client / server architecture with the ability to access system points from the server and system points from the client viewer.

5.4.2 Real Time Data Value Display

The system should support advanced ease-of-use configuration capabilities including drag and drop. Applications that allow a user to select and display a table of points in a separate window without the need for configuring graphic screens is critical for the commissioning and maintenance of the entire system. The user should be able to add points to this table by simply dragging and dropping them from the list of points configured in the system. The values of the points should update dynamically along with a time stamp that indicates when the value changed. In addition, points that meet their alarm criteria should be displayed in a different color.

Double clicking on a displayed point should bring up configuration information on the point including its description and alarm information. Points with Read/Write capabilities should be able to be set from this table.

5.5 Point Cross Reference

The system should support the ability to cross reference where points are used within the system. The cross reference listing should include areas such as –

- Screens
- Point Configuration
- Events and Actions
- Database Logging
- Scripts
- Expressions

The ability to print out the point cross reference information should be provided to assist in documenting the configuration of the system.

6 Project Configuration Wizard

The monitoring and control system should have an integrated project configuration wizard that steps the user through the initial setup and configuration of the system. The wizard should allow the user to select the communication protocols that have been installed and then be capable of detecting Ethernet based PLC devices and OPC servers. Once these devices and servers are detected the user should be able to select the ones they wish to include in the project. The wizard should then allow the user to automatically configure points for a device based on a user specified memory range. The user interface for wizard should be HTML based to provide a simple and easy method for navigation through the configuration steps.

7 Web Capabilities

The software system should also support Thin Client technology to enable graphic screens to be sent to standard web browsers without the need for the user to create HTML pages. The resulting web based graphic viewers should be capable of viewing the graphic screens of the HMI. The web viewers should support the ability to perform setpoints back to the supporting server. Multiple levels of security must be provided including those of the operating system as well as the ability to validate users logging in to the system by user name and password. Privileges granted to web based users should be similar to standard Viewers with the addition to totally disable all set points if required.

Web Viewers should have the capability of being setup in a user mode or broadcast mode. In user mode, individual users are granted an access license when they log into the system. Once the user logs off the system, the license must be immediately available for use by another user. Broadcast mode must also be supported to allow a single license to broadcast the same screen information to an unlimited number of users. In broadcast mode the users will not be allowed to change screens or perform setpoints.

The software system should also have the capabilities of sending and receiving point based information from standard HTML pages. Point information should be capable of driving animated ActiveX graphic objects within the HTML web pages.

The HMI product should provide XML server capabilities to provide data collected from the shop floor out to other applications using the XML format. In addition the HMI should provide the ability to view point and alarm reports in XML format in standard web browsers. In addition, the XML based point and alarm reports should be integrated into a web viewer application that allows the user to view live graphic and alarm screens. Two way interaction with these screens should be provided including control actions by privileged users.

In addition to supporting graphical viewing access via the web, the monitoring and control system should allow for point data to be transferred between servers across the Internet.

The HMI products' web solution must support the launching of 3rd party products from within the client application and for the 3rd party product to be fully interactive within the SCADA's web session.

8 Windows Terminal Services

HMI product should be built to take advantage of Microsoft's Terminal Services product.

Microsoft Terminal Services is a product originally developed for Windows 2003 to deploy a "thin client" solution to deliver Windows sessions to a wide range of desktop and mobile hardware devices. Terminal Services combines the low cost of a terminal with the benefits of a managed Windows-based environment and offers the same low cost, centrally managed environment of the traditional mainframe with terminals, but adds the familiarity, ease of use, and breadth of applications support offered by the Windows operating system platform.

The HMI should support displaying its graphical screens as well as all other runtime and development applications via Windows Terminal Services. Remote diagnostics and system development should be possible through Terminal Services.

9 Applications

The software system should provide a "base architecture" that supports one or more "option" or “application” modules.

9.1 Graphical User Interface/Status Monitoring

The Graphical User Interface should provide a set of tools for graphically representing process status. A graphic editor should be provided to enable creation of graphic screens to represent current process information.

For ease of use, the editor should include cut & paste as well as drag & drop support within a single window and among multiple windows and should include undo/redo support. It should be possible for the user to drag and drop points into a screen from the list of configured points in the system. Support for grouping and ungrouping sets of objects and for readily editing them while grouped are to be included. While editing grouped objects, properties associated with the group shall not be lost during the ungroup/edit process. Object alignment and spacing tools are required so those objects can be properly arranged on the screen.

The editor should include a utility or tool for determining which points are referenced in a screen, which objects reference them, and which points are not currently defined or known to the software. This tool should also include provision to search and replace point names - for both single objects and groups of objects.

A test animation capability should display the screen currently being developed in the runtime environment for rapid prototyping and testing.

The editor software should include the ability to Create/Edit Points from within the Editor. It should also be possible to browse the network to locate computers and projects for available points.

The editing package should include a Wizard / Symbol / Object Library to permit the inclusion of pre-developed or third party graphic objects. Objects contained in the Symbol Library should be objects native to the software solution where appropriate and not simply bitmap representations. The editing package should allow Objects/Wizards to be created with the native graphics and scripting language and added to the Library.

A procedure editor should be included to control setpoints and to perform window management. The graphic editor should include a scripting expression editor to develop application logic.

Graphic objects on these screens can be linked by name to actual device and virtual data through the distributed point database, direct OPC-DA, and Historical expressions in either a real time or historical sense. Objects on the graphics screens can be configured with animation features, causing them to change color and/or position. Text information can be printed to the screen alerting personnel to current point status. Objects should be dynamically scalable - both horizontally and vertically.

The software should support the following dynamic attributes:

- Annunciation, movement, blink, rotation, and fill (uni-directional and bi-directional)
- Gradient fill
- Object border animation
- Object visibility
- Blink fill and blink rate
- Transfer tags for screen transfer or popup windows
- Procedure tags to invoke user defined scripts/programs
- Object and or application help screens
- Alarm information
- Trends charts

- Setpoint tags for point value changes
- Animated frames that can include other graphic objects
- Zoom to Best Fit, Resize Window to Zoom
- Manual and automated rubber band zoom
- Automatic font scaling when changing window sizes
- 1.5 Million Colors
- Alpha Channel Colors
- Graphic objects should include:
- Imported metafile objects
- Embedded OLE, including ActiveX objects, sound, video, clip art, spreadsheets, etc.
- SPC charts
- Trend charts
- Historical Data displays
- Alarm displays
- Arcs
- Lines
- Circles
- Ellipses
- Lines
- Polylines
- Polygons
- Rectangles
- Text strings
- Buttons
- 3 Dimensional Piping creator
- Tag types should include:
- constant - downloads constants to a point
- variable - allows operator input of desired value
- ramp - downloads values in configured increments
- slide - increment/decrement of point values
- toggle - sets digital points to opposite state

Graphics screens should support a Visual Basic or .NET compliant scripting language. Data items and variables can be manipulated by the screen scripting to provide additional functionality in dynamically controlling screen characteristics.

The graphical editor and viewer should be capable of being an ActiveX container. It should be capable of using ActiveX or .NET objects provided with the HMI package or third party ActiveX or .NET controls supplied by others.

The graphical user interface should support ActiveX "methods" to allow the user to interact with ActiveX control objects. Interaction may be through the association of a method to a button or object, which the operator initiates, or methods may be used by the Visual Basic compliant or .NET scripting language for advanced functionality and additional control of the ActiveX components.

The graphical interface should have historical playback and review capabilities. Through a PVR type control interface, the user should be able to select a period of time and then replay the graphical screens and watch the process parameters change on the screen in replay mode.

The graphical runtime should be able to optimize the display using anti-aliasing to provide smooth curves and edges.

The handling of graphic images should be such that they can be scaled without distortion.

The Graphical interface should allow for the creation of large screen sizes that can scale across multiple monitors.

The Graphical UI shall allow the end user to use zoom and pan functions to move around a large screen. The system shall allow for de-cluttering at various zoom levels to increase and decrease graphical detail. These functional shall also be available programmatically to allow for automated zooming and panning as required.

System developers need to have the ability to easily create libraries of re-usable graphics that can be used either embedded or linked on screens. Linked objects need enable the changing a master graphic that can be propagated through an application.

9.2 Menu Navigation

The GUI shall include a single, integrated and comprehensive menu-style HMI navigation system providing the following elements:

- Provide users with organized and structured access to all core system services including:
  - GUI screens in organized user-defined hierarchies
  - Operator services such as alarm and event lists
  - Configuration services such as GUI configuration
- The GUI shall enable users to produce and maintain a single menu system common to the entire GUI to minimize the menu production and maintenance tasks.
- Menu labelling using either or both graphical (icons) and/or text labels. Both icons and text labels shall be user-configurable.
- Hot spot identification, to dynamically and visually indicate to user when their pointing device is above a particular menu item, at least for top level menu items.
- The option to automatically include screen thumbnails as menu icons to assist menu item identification.
- Intuitive, browser like navigation, specifically:
  - Back button - Return to the last screen (or older, from the list of previous screens)
  - Forward - Go to the next screen (or other in the list of later opened screens)
  - Home - Display the home screen
  - Favorites - Each user shall be able to save their own favorite screen(s)
- The menu system must be highly user configurable, without requiring programming or scripting
- The menu system must support the SCADA security system, so that only permitted commands are visible in the menu and available to the logged-on user. This security shall be able to be applied to both main (primary) menu buttons and/or sub-menus.
- Automatic inclusion of new user-developed screens, into user-defined locations within the menu system, based on user-defined screen-naming conventions. For example, the user might wish to define that screens labelled “*trends” are automatically included in a “Trends” location within the menu, while new screens labelled “*_sub” are automatically included within a sub(station) location in the menu.

9.3 Tab Display

The GUI shall provide a graphical container to store and display multiple layered pages of GUI information, providing users with visual tabs to select a particular page or layer for display. The general purpose of this tab display utility being to provide a mechanism to deliver potentially large amounts of information in a compact, structured, repeatable, user-configurable and intuitively-accessible manner. Attributes of the tab display should include:

- User-definable number of tabs
- User-definable tab text and/or graphical (icon) labels
- User-definable tab content, supporting all the same display functions as standard SCADA GUI screens, not limited to but including:
  - Static and dynamic GUI components
  - Alarms, Event, real-time data summaries and other tabular SCADA displays
  - Trends
  - Drawings, diagrams, manuals, images and other 'non-SCADA' information
- User-configurable tab display size, opened by clicking on an element in a standard GUI screen, displayed as a pop-up superimposed on the main display, until being closed by the user. The aim is to provide users with easy access to potentially large amounts of data on specific items, without cluttering their normal displays.
- A highly automated configuration process for replicating multiple instances of tab displays for each instance of repeated I/O device, piece of equipment and/or areas of plant. This automated replication should produce both the SCADA database and GUI elements required to implement every instance of tab display, each being unique to one instance of device, equipment or plant area. Configuration should be reduced to a few simple steps, regardless of how many I/O is represented within the Tab display.
- Ability for users to create their own tab displays for (I/O) devices and equipment from any manufacturer and any industry.

9.4 Data Display

A tabular (list) display of the current value and point attributes of database points shall be provided with the following characteristics:

- This display shall automatically populate with (by default) all points in the database, therefore it shall not require special graphics configuration, beyond being added to one (or more) GUI screens.
- It shall be configurable to display a subset of the complete system by applying filters during configuration time. These filters shall include but not be restricted to a method to filter by:
  - areas of plant
  - data type
  - data point name and/or description
- Comparable filters shall be applicable by users during runtime.
- It shall be possible to control equipment by clicking on a row (individual database point) in the tabular data display, causing the standard control dialog to be displayed for controllable devices. Combined with the auto-configuration capability of this display, this facility shall enable device monitoring and control of the system without the need to first create conventional SCADA graphics. It is expected that this capability will assist during SCADA system configuration, testing and commissioning.
- The data display GUI This list shall be consistent with other SCADA lists and shall include at least:
  - Click & drag column sizing and column positioning

- Quick A>Z / Z>A column sort, plus a multi-parameter sort
- Both configuration-time and run-time filtering, including multi-parameter filter
- Click on a row in the list to display the appropriate operator dialog for that record type

9.5 Attribute Indicators

The GUI shall provide facilities to show the presence of one or many of the following attribute indicators along-side analog and digital indications:

- One of multiple control tag-out (control lock-out) in place
- One of multiple information tags in place
- Alarm disabled status
- Local Force (Manual Overwrite) in place
- Attribute indicators shall have a user configurable look and feel, including:
- Font, size, weight, style, font color, background color
- Indicator shape: Circle, ellipse, square, rectangle, triangle, hexagon, octagon…
- Indicator width and height
- Multiple indicator spacing, orientation (horizontal / vertical)

Attribute indicators shall provide user-configurable tool-tip display to display additional information about the attribute more information when a user moves the mouse over an indicator.

9.6 Operator Control Processes

The GUI shall provide a complete environment for safe and efficient operator control, suitable for operator-intensive SCADA applications. No programming shall be required to deliver the following important functionality:

- A consistent and intuitive control dialog that is displayed when clicking a device to control it.
- Dialog content must be specific and accurate for each type of device and control operation. For example a pump may have ‘run’ and ‘stop’ controls, while a valve might have ‘open’ and close’ commands. This text must be automatically configured based on the device type and not require manual configuration. The dialog must also include text that accurately and specifically identifies the unique device under control.
- Select Before Operate: The ‘Operate’ button should only be highlighted and operable once a specific control command has been chosen, reducing the chance of accidental control.
- Check Before Operate: A final “Are you sure?” type of prompt to operators as a last check before issuing a control. This final check must be able to be disabled if required.
- Alarm control failures: An alarm shall be raised if a control fails to complete after a user-defined time.
- Suppress alarms for operator-initiated changes: The system should distinguish between an operator initiated control and a spontaneous change of state (for example a circuit breaker tripping), thereby preventing the generation of spurious alarms, when an operator deliberately controls a device to a normally alarmed state. Simultaneous Control Prevention
- Prevent multiple users on different computers from controlling the same device at the same time. Once the first operator opens a control dialog, any other operators trying to do the same should be prevented from doing so, and be advised which operator has control of the device and on which workstation.
- Dialog Timeout: The control dialog should automatically close if idle for a user-defined time. This would release the device for control by other operators, thereby preventing any unintentional disabling of control.

9.7 Control Interlocking

The SCADA shall include a control interlocking facility to prevent controls from proceeding when certain conditions exist. The interlock facility shall provide the following services:

- Ability to independently configure unique interlock conditions and outcomes for each controllable state of a device.
- Ability to configure independently for each interlock whether failure to meet all interlock conditions either:
  - prohibits control from proceeding
  - allows the user to override the failed interlock and continue with the control
- The interlock configuration utility must include a modern interlock expression builder, enabling users incorporate any SCADA database values, plus to select operators from a list of available operators, including all common mathematical, logical, and trigonometric operators, along with the following SCADA specific operators:
  - Point is any alarm or warning state
  - Point is in a warning low or high state
  - Point is in an alarm low or high state
  - Point is in a warning low state
  - Point is in a warning high state
  - Point is in an alarm low state
  - Point is in an alarm high state
  - 'NOT' the points alarm ack state
- A facility to display user-defined interlock logic diagrams in common graphical formats to users.
- Users must be able to give each interlock condition a "plain language" name to make it easily understandable to users so that they can identify interlock functional without requiring them to read or understand the interlock logic. An example of "plain language' name is "Feeder 3 must be live" or "All high pressure valves must be closed"
- A facility to display to users any and all failed interlock conditions.
- The ability to import/export interlock definitions for an entire system in CSV and/or XML formats for bulk configuration using external tools.

9.8 Control Tagging

The SCADA must provide a facility to enable users to apply a control tag (software locks) to normally controllable devices. A control tag shall block operation of the device. Requirements of control tagging are:

- A control tag shall prevents both manual (user/operator) controls and controls initiated from within SCADA automation
  - Attempts by a user to control a tagged device shall cause the tag information to be displayed to the user, including Displaying who, when and why a tag was added.
- A control tag user-interface dialog that makes adding, viewing and removing control tags simple and consistent across the system. This shall include the ability for users to select from multiple (at least 50) user-defined control tag types when applying a tag to a device; to define their default tag type; to add a free-format, unlimited-length tag reason message when tagging a device.
- Tag presence should be shown to users by Attribute Indicators
- Adding and removing tags shall be recorded in the Events list
- The Tag should allow support for up to 256 characters

- A 'Control Tag List' to display all tags in a system. This list shall be consistent with other SCADA lists and shall include at least:
- Click & drag column sizing and column positioning
    - Quick A>Z / Z>A column sort, plus a multi-parameter sort
    - Both configuration-time and run-time filtering, including multi-parameter filter
    - Click on a row in the list to display the appropriate operator dialog for that record type

9.9 Information Tagging

The SCADA must provide a facility to enable users to apply an Information Tag to normally controllable devices. An information tag provides an advisory or warning to Users. Requirements of information tagging are:

- When a user clicks on a point, any information tags present are displayed to the user, before the user can control the device. That is, the Information Tag shall act as a pre-control warning, typically of an unusual condition. However unlike a control tag, an information tag shall not prevent the user from completing the control.
- Both Information Tags and Control Tags can be applied to one device.
- When both tag types are present, the Control Tag text message is displayed in preference to the Information Tag message.
- The information Tag display shall include who, when and why a tag was added.
- An information tag user-interface dialog that makes adding, viewing and removing information tags simple and consistent across the system. This shall include the ability for users to select from multiple (at least 50) user-defined tag types when applying a tag to a device; to define their default tag type; to add a free-format, unlimited-length tag reason message when tagging a device.
- Tag presence should be shown to users by Attribute Indicators
- Adding and removing tags shall be recorded in the Events list
- An 'Information Tag List' to display all information tags in a system. The list shall provide both sort and filter capabilities consistent with other lists in the system. Users shall click on a row (single entry) in the Information Tag list to display the same Information Tag dialog used to view/remove the control tag.
- Tag presence is shown by Attribute Indicators
- Adding and removing tags is recorded in the Events list

9.10 Local Force

A 'Local Force' facility is required to enable users to write a current value for a point into the SCADA database, and have the SCADA ignore telemetered readings for that point. This is required when a transducer is not yet installed, is faulty, or has been removed for servicing. General requirements:

- Local Force is required for both analog and single-bit digital values
- The presence of a Local Force shall be shown by Attribute Indicators
- Local Force values shall be displayed in an identifiable Local Force color
- Adding and removing a Local Force shall be recorded in the Events list
- A Local Force list shall be provided that displays all points in a system where a Local Force is in place. This list shall be consistent with other SCADA lists and shall include at least:
    - Click & drag column sizing and column positioning
    - Quick A>Z / Z>A column sort, plus a multi-parameter sort
    - Both configuration-time and run-time filtering, including multi-parameter filter

- Click on a row in the list to display the appropriate operator dialog for that record type

9.11 Alarming

The software system must support an Alarm Management module capable of alarm annunciation and routing capabilities. The alarm text associated with each alarm should be user configurable.

Alarms are to be applied as follows:

- Digital Points - the alarm generating condition (0 or 1) should be selectable.
- Analog Points - the alarm generating conditions should be evaluated based on alarm criteria selected:
- **Absolute** - There should be two levels of high alarming, HI-2 and HI-1, and two levels of low alarming, LO-1 and LO-2. HI-1 and LO-1 are also known as warning alarms. For high alarming, an alarm should be generated when the point value reaches or exceeds the value specified for HI-1 or HI-2. For low alarming, an alarm should be generated when the point value reaches or falls below the value specified for LO-1 or LO-2.
- **Deviation** - Alarm limits for deviation alarms should be given in positive values. The HI-2 and HI-1 alarms should be generated when the difference between the current point value and the Deviation Point value is positive and reaches or exceeds the specified limits. The LO-1 and LO-2 alarms should be generated when the difference between the current point value and the Deviation Point value is negative and the absolute value of the difference reaches or exceeds the specified limits.
- Rate of Change - Rate of Change alarms should be provided to detect either a faster or slower than expected change in the value of a point.
- Duration - The Alarm Display should include total time in alarm state.

Alarms should be configurable to be filtered and asynchronously sent to users based on user role and scope of responsibilities. Alarms should be configurable with respective priorities, divided into classes, and color-coded for display. There should be user-defined logging criteria, user-defined acknowledgment and deletion criteria, user-specific textual messages and operator help text. The alarm list can be toggled between dynamic and static display and quickly filtered to limit the current view to a particular alarm set of interest.

The system should support "alarm blocking". Users should be able to define an alarm hierarchy and block the generation of "lower level" alarms if a "higher level" alarm is present. This allows for operators to concentrate on primary causes rather than receive all the resulting secondary problems. For example, if a conveyor stops then all machines feeding it would also stop. The operator needs to determine why the conveyor stopped - the operator does not need to see the other alarms. In this example, fixing the conveyor will fix those alarms as well.

The system should provide for an automatic routing of configured alarm messages to display type pagers. The routing should be configurable as to personnel or pager ID receiving the message. It should be possible to upgrade the paging system to support dual outputs - allowing messages to be sent to local pagers, or to a dial-up paging system.

9.11.1 Alarm user interface

Alarm user interface requirements shall include at least:

- An alarm list that is consistent with other SCADA lists and shall include at least:
  - Click & drag column sizing and column positioning
  - Quick A>Z / Z>A column sort, plus a multi-parameter sort
  - Both configuration-time and run-time filtering, including multi-parameter filter
  - Click on a row in the list to display the appropriate operator dialog for that record type
- Ability for users to acknowledge a single alarm and/or acknowledge all alarms on the current screen

- A facility for users to mute / unmute the alarm horn
- User facility to change alarm limits, with a system wide ability to enable/disable this feature, plus the ability for users to permanently record the reason for changing alarm limits in the Events list
- User facility disable / enable alarms, plus the ability for users to permanently record the reason for disabling an alarm in the Events list
- An "Off-Normal" List that displays all points not in their normal state, regardless of alarm conditions. The Off Normals list user interface shall be consistent with other SCADA lists

9.12 Event Recording and display

The SCADA shall include a comprehensive and permanent event recording system, including recording, display, filtering, exporting and printing of system and operator activity, including the following activity:

- Alarms
- Non-alarm changes of state
- Sequence of Events (SOEs)
- Operator actions:
  - Control
  - Interlock acceptance
  - Control & Information Tag, apply & remove
  - Apply & remove Local Force (Manual Overwrite)
  - Add Note
  - Operator comments about SCADA events
  - Operator event messages
- The Events list shall be consistent with other SCADA lists and shall include at least:
  - Click & drag column sizing and column positioning
  - Both configuration-time and run-time filtering, including multi-parameter filter
  - Click on a row in the list to display the appropriate operator dialog for that record type

9.13 Operator Notes

There shall be a general purpose notes recording/viewing facility for users to record and share information. The following are minimal requirements:

- The systems shall be configurable to automatically provide a unique note for device and/or screens without individual configuration.
- The notes text editor shall support SCADA security by NOT providing access to SCADA disk drives or other data storage devices during operations like Save, Save As and Open.
- Editing and saving a note shall automatically create a Notes history, that is each 'Save' after editing shall preserve the previous revisions of any note. These earlier revisions shall be immediately accessible to users through a 'Notes History' interface that shall be integral to the Notes editor.
- Devices Notes shall be easily accessible through the device dialog or similar. Screen notes shall be easily accessible through say a 'Notes' button on the screen.
- The system shall automatically maintain a List of all current Notes (those containing user entered information). This list shall be consistent with other SCADA lists and shall include at least:
  - Click & drag column sizing and column positioning
  - Quick A>Z / Z>A column sort, plus a multi-parameter sort

- Both configuration-time and run-time filtering, including multi-parameter filter
- Click on a row in the list to display the appropriate operator dialog for that record type

9.14 Command/Scripting Language

The scripting language used by the system must be like VB basic but is not the sourced Microsoft's Visual Basic for Applications product In addition the scripting engine must support .NET and Python scripts to be ran side by side basic scripts as to ensure the customer can choose which scripting language that best applies or a mixture there of. Scripts can be simple or complex and allow users to automate operator tasks, and create automations solutions. Scripts must be capable of running in either the configure environment ("draw" or "edit") or the run environment ("view"). The .NET scripting language must use Microsoft's IntelliSense feature, exposing all properties, methods and events of graphic objects. This includes 3rd party ActiveX controls.

Python scripting language must provide IntelliSense feature, exposing all properties, methods and events of graphic objects. This includes 3rd party Python libraries. Python scripts shall be editable by default in a full-featured script editor based on Visual Studio Code.

Editing of the basic scripts will be within the HMI product itself and the .NET scripting is done within or with a 3rd party editor.

9.15 Data Logging

Data collected by the software system should be logged via ODBC into a relational database to support historical reporting and analysis, or to a dedicated Historian. The system should support multiple SQL compatible databases and/or formats. Configurable logging of points, alarms, and events should be supported without forcing the application developer to understanding database internals. Custom application software must not be required to log data. Configuration of the logging characteristics of a point should automatically configure the database that should store the data. A variety of database management systems should be available for use.

The system shall provide an option for integrating Microsoft SQL Server into the application as the standard data-logging database. As an integrated option, it shall have an icon as part of the configuration environment to easily launch the SQL application.

The software system shall support a high-speed data-logging rate to the Microsoft SQL Server database though a "bulk insertion" method. Bulk insertion reduces database overhead by storing information temporarily in memory and writing a larger volume of information to the database at a single time.

The software system must also support high speed / high volume logging and retrieval. This capability is often referred to as a Data Historian.

Point and alarm data is to be logged upon a "trigger" event. The following triggers for logging point and alarm data are required.

Point Data	Alarm Data
At Time of Day	On Generation
On Time Interval	On Reset
On Point Update	On Acknowledgment
On Event	On Deletion
Gated Based on Logical Expression	

Point attributes, which should be available for logging, include

- Point Value
- Previous Value
- Raw Value
- Alarm State
- Resource
- Time Last Logged
- Engineering units

The logging module should support the logging of multiple point attributes into a single record based on a single trigger. The logging module should also support logging multiple points and their attributes to a single record. The logging module should support the simultaneous logging of multiple tables of data consisting of combinations of single points, multiple points, alarms, and events. This supports the creation of custom database tables unique to an application.

Through configuration alone, the logging utility should support "store and forward" to selected database management products. Data should be buffered on the node collecting it and automatically forwarded to the node where it is to be logged. During a communications outage between the two nodes, the data collection node should continue to buffer data. Upon restoration of communications, the data collection node should automatically forward the buffered data to the logging node for storage.

User configurable database maintenance actions, which are executed automatically, based on database size or number of records should be supported. Examples of these actions include exporting data to a CSV file then purging the records from the database. No custom software should be required to implement this support.

Reports using the logged data may be generated using standard third party database management tools such as spreadsheets and report writers.

9.16 DDE Client and Server Interface

The software system must support a DDE interface in both client and server modes. In the server mode, DDE aware applications (clients) such as Microsoft Excel™ should be able to access data managed by the Monitoring and Control System. The DDE client interface support the use of third party applications to monitor, analyze, report, and modify point data.

Required Services for the HMI server support include:

- Request Point Configuration Data (e.g. alarm limits, engineering units labels)
- Request Point On-Change (DDE Peek)
- Request Point Update (DDE Poke)

In the client mode, DDE server applications such as device communications drivers should be able to act as a source of data to the HMI, for both reading and writing of data down to factory floor devices or external systems. For maximum flexibility in selecting third party servers and for maximum software performance, the HMI must support both the DDE and AdvanceDDE protocols.

9.17 OPC Client and Server Interface

The software system must support an OPC UA, OPC DA and OPC Alarm and Events (A&E) interface in both client and server modes. In the server mode, OPC aware applications (clients) should be able to access data managed by the System. The OPC Client interface should support the use of third party OPC Device Communication Servers. The OPC Server interface should allow for point information collected and maintained by the software

system to be sent to OPC applications requesting the data. The OPC UA server must also expose the classes and objects (Model) if it is configured in the SCADA.

The OPC DA and OPC UA client should allow for browsing of available OPC servers and the published points.

For OPC UA based devices, the developer will have the ability to select objects or data in the OPC UA Server and allow the ability to automatically create the data points for the SCADA configuration.

9.18 Data Trending

The trending module should be capable of supporting one or more embedded trends within the runtime graphics user interface. The following types of trends should be supported:

- Trends with Multiple Y Axes
- Trends with Multiple X Axes
- Trends with multiple time periods
- Reference curves
- XY Plots

The ability to select any numeric value or values being displayed on a graphic screen and quick trending the values should be provided. There should be no need to have pre-configured graphic screens created to display the quick trends.

The trending option should support display of an unlimited number of pens on a single trend chart. Each pen should display either dynamically updating data or provide seamless access to historical values based on user request. In addition, user should be able to compare data from different time periods.

Trending should support the creation and re-display of files with reference data from the currently displayed trend. This export capability should produce "CSV" format files so as to be compatible with standard office automation tools such as spreadsheets and databases.

Users should be able to analyze trend data by scrolling through time, changing the range for point displays, zooming into an area of the trend and selecting a new time period to display.

Printing of trend charts is required.

The trending module should support reading data from "CSV" files for the display of data collected or generated by other applications within the software system framework.

Trend layout should be highly configurable including colors, tick marks, legends, title, and fonts. The update rate for data being displayed from log files should be configurable. The data being trended should be configurable. This configuration must be modifiable at runtime without requiring a development system or license.

Trend data can be supplied from multiple sources including current point data, data from .CSV files, and data logged to a database. Data displayed by the trend manager should support time-based or on-change based sampling.

Trending should support the display of array points. Arrays can be interpreted as independent variables or as a time series of a single variable. The latter case supports buffering of data in high speed sampling applications.

Trending should support rapid retrieval of a large time span of data and be able to aggregate the display of the data into the time axis of the chart. It should also be able to handle aggregating functions such as the min, max, and average display of values over the time period as well.

9.19 Statistical Process Control

Data collected though the Monitoring and Control System should be accessible by a Statistical Process Control Module. This includes data from plant floor devices, manual data entry, custom applications, and flat files. SPC Charts/Reports should be provided as ActiveX objects which can be dropped into graphic screens. The software should support an unlimited number of quality characteristics or sub-groups. The quantity of data displayed in SPC charts should be user configurable and should be updated dynamically as new data is collected. SPC ActiveX objects should allow the user to dynamically switch between a chart and report view of the data without the need to reconfigure screens. Users should be able to scroll through data that is not initially displayed on a chart. Standard Charts/Reports should include:

- Xbar & Range
- Xbar & Sigma
- X Individual
- Histograms
- Pareto
- U
- P
- C
- nP

Software should support the following control checks where "N" is configurable. Each may be separately enabled or disabled.

- Any one subgroup beyond upper or lower limits
- N subgroups on the same side of the center line
- Trend of N consecutive subgroups in a row up or down
- N subgroups in a row alternating up and down
- Sets of Subgroups in a row within one, two or three standard deviation

All SPC Out of Control (OOC) conditions should be capable of being processed by the Monitoring and Control System to:

- Change the color of a graphic object to red or flashing
- Sounding an audio alarm
- Logging the OOC condition to a device or file
- Send an alarm message to a user or another process

Individual Quality Characteristics are configurable based on the following parameters:

- Enable or disabled
- Subgroup Size and Interval
- Center Line, Upper and Lower Control Limits
- Control Limit Recalculation Type (Manual, Automatic)
- Control Checks and Alarming

Configuration information should be centralized on a server node and support SPC charts operating on viewer nodes. All nodes displaying an SPC chart must have the same control limits.

Users should be able to dynamically modify control chart configuration parameters through the user interface. In addition, they should be able to delete outliners (points that are out of reasonable range) prior to recalculation of control limits.

SPC charts may be embedded in graphic display screens as ActiveX objects to provide seamless integration among applications.

SPC charts will also need to allow annotations that can be shared for insights between users.

9.20 Classes and Objects based Configuration environment

The screen and database environment should also allow for a structure configuration method using object oriented design. This should allow for the creation of classes (object types) from which objects can be derived and various parts of the application created automatically. This would include point definition, alarm configuration, script, graphical mimics and screens. The creation of templates needs be supported to allow for easily re-using configuration across multiple applications.

9.21 Production Tracking

The software system should support a configurable Production Tracking application that monitors point values and maintains an accurate image of work-in-process inventory. The Production Tracking module should be configurable with respect to the layout of plant conveyor systems, the type of items being tracked (serialized vs. non-serialized), and the type of plant floor inputs available for tracking. Example inputs include radio frequency (RF) tags, bar codes, type detectors, and limit switches. The tracking system should be auto correcting based on collected data.

Each item should have a set of attributes including location, status, and user-definable fields. Attributes such “product build” options and/or inspection status should be displayable and modifiable through point values.

The Production Tracking application should support manual transactions to add, delete, modify, move, re-sequence, and scrap items.

The Production Tracking module should include a comprehensive set of routing and control functions. These functions should interface to the HMI's data collection subsystems to exchange data with factory floor devices. The routing and control functions should be support user programming and configuration. The functions should operate based upon the work in process image maintained by the Production Tracking module.

The primary function of the tracking system is to provide tracking data to plant floor users and applications. As such the data maintained by the production tracking module should be accessible through the graphical user interfaces and through a complete set of application interfaces. Additional applications such as dynamic scheduling, material delivery, and assembly broadcast depend on tracking data.

The Production Tracking module should support operation in the distributed software system environment.

9.22 Equipment Control

The software system should support an Equipment Control module which includes the capability configure logic and schedule events based on the following criteria:

- Time of day
- Time Interval
- Production event
- Point change
- Alarm Generation

In response to configured events, the system should invoke configurable actions that include:

- Log Event
- Acknowledge Alarms
- Enable / Disable an Alarm
- Recipe Upload / Download
- Execute a Script
- Set a Point Value
- Copy a Point Value
- Execute a Procedure

Scripts should be modular and re-usable. Scripts should be in a Visual Basic compliant language.

An event editor should be included to permits users to configure, monitor and debug applications. Users should be able to monitor the progress of control programs and dynamically modify their operating characteristics. The event editor should at least support these debug tools:

- Run in Single Step Mode
- Start/Stop Scripts
- Set Break Points
- Step into or Over Sub-Scripts
- Watch variables change

Control programs should have access to all of the application programming interfaces (API's) within other Monitoring and Control applications such as DDE interfaces, equipment/device interfaces, production tracking, and system alarming and logging.

9.23 Recipe Management

A module that is capable of monitoring and managing equipment recipes should be available. The Recipe module should support both batch and discrete part processing requirements. The Recipe module should support dynamic configuration of recipe information.

The Recipe management module should support the creation and management of device independent recipe data for production processes. Configuration data should be stored in recipe groups to simplify information management. Within a group, recipe parameters should only have to be entered one time to minimize duplicate data entry.

Parameter values should be supported for each recipe. The module should support the concept of device independence. This should permit the configuration of a recipe a single time yet also permit downloading that same recipe to different devices, i.e. different programmable controller models or manufacturers. This device independent support no longer requires the maintenance of separate recipes for the various pieces of production equipment in the facility.

There should be a dedicated user interface utility. The utility should support the display of recipes, parameters and the mapping of parameters to devices. Ideally, this display should support a spreadsheet-like interface for ease of use and to help minimize user training. Like a spreadsheet, the information display formatting should be user configurable. The interface should follow the Microsoft Windows look & feel and include tools such as copy/cut/paste.

Overall, the user interface should support the ability to:

- Create and manage recipe parameters, recipes, and maps side-by-side in a spreadsheet format.

- Import and export recipe groups from/to CSV format files.
- Archive recipe groups.
- Automatically reconcile recipe groups to accommodate changes in the group's structure and layout.
- Compare recipes.
- Validate recipe information.
- Manually upload recipes.
- Manually review/modify parameters and download recipes.
- Create recipe parameter files, to support automatic upload and download of recipes.
- Identify a Batch ID for recipe downloads.

For maximum flexibility, the Recipe management module should also support a graphic user interface. Recipe objects should be implemented as OCX controls that may be embedded in graphic screens. These objects should let a run-time user:

- Manually upload recipes.
- Manually review/modify parameters then download recipes.

To support the development of sophisticated applications, the Recipe management module should support an interface to the software system's scripting capabilities. This scripting interface support should include the ability to:

- Automatically upload and download recipes based on system events, such as point changes from a shop floor device.
- Import and export recipe groups from/to CSV format files.

The module should support a wide range of data types including:

- Signed integers
- Unsigned integers
- Digital/boolean values
- Real numbers
- Text strings

9.24 Calendar Based Control

The software system should provide for calendar based control operations through a self-contained graphic users interface. The software system should allow you to dynamically create, maintain, and execute a calendar schedule of manufacturing events and corresponding actions. For example, a system should allow the user to perform such actions as turn on lights, heat, and equipment based on a schedule which they configure and maintain through simple point and click actions.

Configuration of the calendar-based control should allow the user to define different types of days – production, weekend, holidays, to conform to 4 day or 5 day work weeks, etc. Users should then be able to configure manufacturing control events they want to occur on a particular type of day and the time they should occur. Association of days on the calendar with a particular type of day should be through a point and click interface. The graphic user interface for the calendar-based control should allow the user to dynamically change or override the established schedule.

The graphics interface for the calendar-based control should be available on both Servers and Viewers.

9.25 Historical Data Display

The system should provide for tight integration with Historian and with relational databases through ODBC interfaces. Once data has been logged to the database, open system tools should make the extraction and sharing of the data easy between users and applications.

9.26 Integration APIs

The software system should have a set of open interface APIs (Application Program Interfaces). The APIs should include –

- Point Management API – Allows third party programs to be interfaced to the software system and pass point data to and from the system in real time fashion.
- Alarm Management API – Should allow alarm information to be passed to and from the API.
- Logon API – Should allow the user to create their own custom logon dialog boxes.
- Device Communication Toolkit – Should allow the user to construct communication modules to third party devices with published protocols.

9.27 Marquee Driver

The Software system should have the ability to send alarm information and messages to multi-line marquee display devices. A Marquee module should be available fully configurable. No custom coding should be required to retrieve alarm or message information. It should be possible to determine which set of alarms you want to send to the actual marquee device.

Marquee configuration should include:

- Marquee Types - All marquee devices that use the same header, footer, message wrap, empty message, and attributes belong to the same Marquee Type.
- Marquee Ports - The PC communication port or the network communication port to which one or more serial marquee devices are connected. If more than one device is on a port, the header and/or footer information (from the Marquee Type) should indicate which device on which the message is to be displayed.
- Marquee Devices – The ability to assign a Marquee ID, Marquee Type, Marquee Port, header, footer, empty message, and display time for each marquee device within a project should be provided. The option of associating sets of marquees into Marquee Groups should be provided.
- Marquee Messages - For each Marquee message, it should be possible to assign a message ID, Alarm ID, alarm state, message header, message footer, message text, the marquees on which to display the message, the attributes associated with the message, and any HMI point values to be displayed with the message.

9.28 Alpha-Numeric Alarm Paging/Notification

The Software system should provide advanced communication capabilities such as the ability to send process alarms to standard alphanumeric or numeric pagers carried by a mobile workforce. The paging capabilities should be designed on a client / server architecture of to avoid duplication of configuration information. Paging capabilities should support standard IXO/TAP protocols used by common paging systems. Paging should also support email and SMS notification systems.

Paging/Notification features should include:

- On line configuration of users and paging numbers.
- Enable or disable users from receiving pages.

- Escalation of pages
- Filtering of pages based on point groupings, alarm classes, or alarm IDs.
- Dynamic on-line configuration changes.
- A scripting interface for automatically sending pages, changing a user's pager number, or disabling a page.
- Customizable pager messages.
- Configuration templates for fast setups.
- Support of distribution lists.

9.29 Redundancy

The principle of redundancy in automated systems provides for switchover of functionality to a backup component in case of failure of a primary component. The switchover is considered automatic if no operator intervention is required. Redundancy applies to both hardware and software, and implies minimal loss of continuity during the transfer of control between primary (active) and redundant (backup) components. Redundant systems reduce single points of failure, preventing loss of functionality.

The system must support a software based redundancy solution as well as hardware dependent, high availability architectures as well.

For cell control systems, the major levels of redundancy include:

- PLC
- PLC LAN or serial connections to server
- Computer networks
- Computer

Each level of redundancy provides a failover system that allows continuous system activity with minimal loss of data. The following sections briefly describe each level.

9.29.1 PLC Redundancy

The system should support PLC redundancy. PLC redundancy lets control transfer from a primary programmable controller to a redundant one in case of failure. When the primary PLC comes back on line, control can be transferred from the redundant PLC back to the primary with minimal loss of data. The redundancy can be synchronous or independent. Synchronous systems coordinate control and handling of data between CPUs of the active and backup units, while in independent systems each PLC acts like an active unit and is not constrained by the others.

9.29.2 Cabling Redundancy

The system should support Cabling Redundancy. Cabling redundancy involves separate physical connections to the same device. The devices can be on a LAN or may require serial connections. Redundant cabling provides an alternate communication path to the device if the association with the host computer is lost due to failure of the primary path. The implementation of cable redundancy with respect to host monitoring/control systems differs with the device protocol involved.

9.29.3 Server Redundancy

The system should support Server Redundancy. Server redundancy involves a primary factory monitoring server and a redundant "Hot Standby" server. The redundant server is essentially a mirror image of the primary server, running alternate monitoring/control processes and applications. Data collection is performed via independent or shared network paths to the same devices, depending on the protocol. The characteristics of the selected communications protocol(s) determine the details of the configuration.

Upon detection of failure of the primary server, the secondary server can assume control of data collection, alarm functions, applications, and allow user access with minimal loss of continuity. When the primary server comes back on line, control can be transferred back, and the secondary server will resume its backup role.

9.29.4 Computer Network Redundancy

The system should support Computer Network Redundancy. Computer network redundancy is similar to cabling redundancy, except it covers computer to computer communications rather than computer to programmable controller. Computer network redundancy provides an alternate network path in case of failure of the primary network.

9.29.5 SQL Always On

For application that leverage Microsoft SQL Server, the software needs to support SQL server with Always On.

9.30 System Monitoring

The software system should provide system monitoring to allow the user to monitor the most important part of the system – the computer. System monitoring should monitor both the software system processes as well as key Windows operating system and network parameters. System monitoring should be able to monitor the status of the computer the software system is running on as well as other Windows computers on the network. System monitoring should allow for key system parameters and statistics such as alarm frequency, device communications, data collection and throughput, inter-process communications, data logging, point management, and user registration to be monitored and recorded. System monitored information can be alarmed on.

9.31 CNC Ethernet Connectivity

The system shall have the capability of connecting to and collecting data from Fanuc CNC controllers over Ethernet as well as High Speed Serial Bus (HSSB) interfaces. The data which is collected shall be capable of being logged or displayed as standard point information.

9.32 DGR Screen Playback

The system should provide the capability of playing historical logged data back through the graphic screens so users can review past situations to analyze what might have caused a problem. This feature should provide a wide range of playback speeds with start and stop capabilities so the user can quickly isolate a particular period of time they are interested in. In addition this functionality shall have the ability to affect the entire interface or only specific elements enabling the user to view mixed real time and historical information on a single screen.

10 Electronic Signatures and Electronic Records

The system should be able to handle the addition of electronic signatures to any point or alarm. Electronic signatures should include one or two signatures per any runtime change including alarm acknowledgement. User accounts should have the ability to be configured within Windows security and a period of continuous use should be configurable were one token (password) would need to be entered. There should be a provision to force 2 tokens even during this period of continuous use. Users should have the ability to enter comments at time of signing and the system should allow for comments to be selected from a predefined list or entered in a free from text box. No changes should occur until the authentication of the user is verified and an electronic record is created about the transactions. This record must contain the time, date, full name of the perform by , full name of the verified by, the original value and new value, the description of the action and any comments entered at time of signing.

11 Monitoring and Control Checklist

The following list contains the key items that need to be considered when selecting a software system.

Feature	Supported – Yes / No
STANDARDS AND TECHNOLOGY	
Open System Design	
True Client / Server Architecture	
Enterprise Server Support	
Viewers (Without the need to reconfigure point databases)	
Web Based Thin Client Viewers – Internet Explorer and Netscape	
Web Viewer User and Broadcast mode licensing	
Operating Systems Support - Windows NT, 98, and 2000	
Microsoft Technologies Support – (ActiveX, ODBC, OPC, DDE, COM/DCOM, XML)	
Industry Standards Support – OPC, OPC UA, TCP/IP	
Object Technologies – ActiveX and user created	
Web Technologies – Thin client and html based	
Windows Terminal Services Support	
Wireless Technologies for Viewing and Entering Information	
Application Programming Interfaces	
Multi-threaded scripting	
REAL TIME DATA MANAGEMENT	
Data Sharing without duplicating configurations	
Required Data Types – Discrete, float, analog, string, global, arrays, structures	
Device Communications – capable of supporting hundreds of devices and protocols.	

System Points – Users, Roles, Time, Date, Alarm counts, etc.
Real time tabular display of point values without the need to create screens.
Point Cross Reference indicating where points are used in the system
DIVERSE APPLICATION MODULES
Graphical user interface with status monitoring
Web Viewer
Thin Client Windows CE Support
Alarming
Data Logging
DDE Client & Server Interface
OPC UA Client & Server Interface
Data Trending
Statistical Process Control
Production Tracking
Equipment Control
Recipe Management
Calendar Based Control
Historical Data Display
Integration APIs
Marquee Driver
Alpha-Numeric Alarm Paging
Redundancy
System Monitoring

12 Vendor Requirements

12.1 Development Life Cycle

1. The vendor must have an established development life cycle that allows for traceability of features and functions throughout that life cycle.
2. The vendor must have a formal and documented set of quality assurance procedures that are applied to the engineering design, development, and documentation of the software. The presence of a formal quality assurance department shall be required.
3. The vendor must also demonstrate that its source code for the product is regularly archived both on-site and off-site in facilities suitable to withstand physical harm.
4. The vendor shall allow for on-site auditing of the development life cycle to ensure good practice.

12.2 ISO 9001 certified

1. The vendor must be able to demonstrate that it has established procedures.
2. Vendor needs to be certified under the ISO 9001-2001 guidelines.

12.3 Secure Architecture and Certification

A secure reference architecture and guidelines shall be provided by the vendor to provide optimal security architecture and configuration. The product must also be Achilles certified for Achilles Practices Certification (APC) to meet IEC-62443-2-4 standards.