

Spirit^{IT} eXLerate

Measurement supervisory software



- Application basics
- Communication
- Tag database
- HMI displays
- Alarm management
- Trending
- Intervals, periods & events
- Reports
- Databases

Measurement made easy

Spirit^{IT} eXLerate
Human Machine
Interface

Introduction

Welcome to the exciting world of Spirit^{IT} eXLerate!

Using Spirit^{IT} eXLerate, you can create your complete real-time HMI applications.

There are two reference manuals:

- The ‘Installation manual’, with the system installation and setup guide.
- The ‘Configuration manual’ describes how to create a full-featured, real-time HMI application using Spirit^{IT} eXLerate.

For more information

All publications of Spirit^{IT} eXLerate are available for free download from:



Search for

eXLerate Installation manual	IM/eXL-EN
eXLerate Configuration manual	CM/eXL-EN
Flow-X function reference manual	CM/FlowX/FR-EN
eXLerate release notes	RN/eXL-EN

Contents

1.	Introduction	3		
1.1.	Features	3	4.2.1.	Protocol table.....
1.2.	Application examples.....	3	4.2.2.	Query Table.....
1.3.	Cyber security.....	3	4.2.3.	Advanced Read mode.....
1.4.	Manuals	4	4.3.	Update device values
1.5.	Target audience.....	4	4.3.1.	Communication settings.....
1.6.	Document conventions.....	4	4.3.2.	Display editing.....
1.7.	Abbreviations.....	5	4.3.3.	Tag Database updates
1.8.	Terms and definitions.....	6	4.3.4.	Visual Basic updates.....
2.	eXLerate applications	8	4.3.5.	Advanced Update mode
2.1.	Application Shortcuts	8	4.4.	xlConnect
2.2.	Open an application	8	4.4.1.	Logging and debugging
2.3.	Create new application.....	8	4.4.2.	Flow-X communication.....
2.4.	Save an application.....	8	4.4.3.	Modbus communication
2.5.	Close an application.....	9	4.5.	OPC Server.....
2.6.	eXLerate user accounts	9	5.	Tag Database
3.	Application basics	11	5.1.	Tag naming & reference fields.....
3.1.	Application modes.....	11	5.2.	Communication fields.....
3.1.1.	Design mode.....	11	5.3.	Values fields
3.1.2.	Preview mode	11	5.4.	Alarm fields.....
3.1.3.	Verify mode	11	5.5.	Trend fields.....
3.1.4.	Runtime mode	11	5.6.	Periodic fields
3.1.5.	Switching modes	11	5.7.	Tag count
3.2.	Excel basics	12	6.	Calculations.....
3.2.1.	Worksheets.....	12	6.1.	Calculation sheets.....
3.2.2.	Cell data	12	6.2.	Store values
3.2.3.	Shapes, pictures and charts.....	12	7.	Displays
3.2.4.	Naming cells and shapes.....	12	7.1.	Configuration tables
3.2.5.	Formulas	13	7.1.1.	User table.....
3.2.6.	Visual Basic for Applications	14	7.1.2.	Worksheet table
3.3.	eXLerate worksheets	14	7.1.3.	Style table
3.3.1.	■ Communication sheets.....	15	7.1.4.	Color table
3.3.2.	■ Internal sheets.....	15	7.1.5.	Button table
3.3.3.	■ Calculation sheet	15	7.2.	Display sheets.....
3.3.4.	■ Configuration sheets.....	15	7.2.1.	New display sheets
3.3.5.	■ Displays.....	15	7.2.2.	Text, live values and units.....
3.3.6.	■ Reports	15	7.2.3.	Charts
3.3.7.	■ Safety precautions	15	7.2.4.	Pictures
3.4.	eXLerate wizards.....	16	7.2.5.	Shapes
3.4.1.	Tag & Object wizard	16	7.3.	Animations
3.4.2.	Calculation wizard	16	7.3.1.	Animation object names
3.4.3.	Button wizard	16	7.3.2.	Configuration.....
3.4.4.	Color wizard	17	7.4.	Buttons & navigation.....
3.4.5.	Language wizard.....	17	8.	Alarm management
3.4.6.	eXLerate engineering tools	17	8.1.	Defining alarms
4.	Data communication	19	8.2.	Alarm groups.....
4.1.	Communication model	19	8.3.	Active alarms.....
4.2.	Set-up communications	19	8.4.	Historical events.....
			8.5.	Advanced alarm usage.....

9.	Trending.....	41	14.3.2. SQL queries on worksheet.....	64
9.1.	Defining trend tags.....	41	14.3.3. SQL queries in VBA.....	65
9.2.	Data storage.....	41	15. Terminal Services.....	67
9.3.	Display trends.....	41	15.1. Requirements and licenses.....	67
9.3.1.	Trend Chart.....	42	15.2. Configuration.....	67
9.3.2.	Trend Pen Selector.....	42	15.3. Using terminal services.....	67
9.3.3.	Trend Navigator.....	42	16. Multi-lingual systems.....	69
9.4.	Advanced trend functions.....	43	16.1. Windows language packages.....	69
10. Editing values.....	44	16.2. Multi-lingual application.....	69	
10.1.	Allowing user input.....	44	16.2.1. Language sheet.....	69
10.2.	Editing table.....	44	16.2.2. Multi-lingual tag database.....	70
10.2.1.	Configuration.....	44	16.2.3. Language selection.....	70
10.2.2.	Group-wise editing.....	45	16.2.4. Multi-lingual displays and forms.....	70
10.2.3.	Edit lists.....	45	16.2.5. Multi-lingual buttons.....	70
10.2.4.	Date /time editing.....	46	17. Daylight saving time.....	71
10.3.	Runtime editing.....	46	18. Document revisions.....	73
11. Interval periods and events.....	47	Appendix A. Troubleshooting.....	74	
11.1.	Interval table.....	47	A.1. Do's and don'ts.....	74
11.2.	Periodic data.....	47	A.2. Communications.....	74
11.2.1.	Weighted averages.....	48	A.3. Reports.....	74
11.2.2.	Latch values.....	48	A.4. Windows event viewer.....	75
11.2.3.	Latch average values.....	49	A.5. Diagnostic information.....	75
11.3.	Calculation triggers.....	49	A.6. Performance monitor.....	75
11.4.	Reset historical values.....	50	Appendix B. Constants.....	76
11.5.	VBA events.....	50	Appendix C. License model.....	77
12. Reports.....	51			
12.1.	Design vs. runtime.....	51		
12.2.	Report table.....	51		
12.3.	Report design.....	52		
12.4.	Report generation.....	52		
12.5.	VBA report functions.....	53		
13. Redundancy.....	54			
13.1.	Redundant communication links.....	54		
13.1.1.	Full redundant communication.....	54		
13.1.2.	Validity check redundancy.....	54		
13.2.	Redundant devices.....	55		
13.2.1.	Separate device tags.....	55		
13.2.2.	Combined device tags.....	55		
13.3.	Redundant supervisory.....	56		
13.3.1.	Servers.....	56		
13.3.2.	Clients.....	56		
13.3.3.	Configuration.....	56		
13.3.4.	Application development.....	57		
13.4.	Communication routing.....	60		
14. Databases.....	62			
14.1.	The system database.....	62		
14.1.1.	The embedded system database.....	62		
14.1.2.	The external system database.....	62		
14.1.3.	System database tables.....	63		
14.2.	Foreign databases.....	63		
14.3.	Application databases.....	64		
14.3.1.	User definable tables.....	64		

1. Introduction

Spirit^{IT} eXlerate is the supervisory software package of ABB. With Spirit^{IT} eXlerate, you can create full-featured, real-time HMI applications, using a well-known, user-friendly spreadsheet environment.

A Spirit^{IT} eXlerate application gives the operators a robust and complete visualization and control of the process.

1.1. Features

Spirit^{IT} eXlerate has the following functionality:

- Full-featured real-time HMI software;
- Made for oil & gas systems;
- Acquire real-time measurement data from field devices, such as flow computers, logical controllers, gas chromatographs, utilizing different communication protocols like Flow-X Client, Modbus, OPC;
- Display and monitor measurement values and equipment status, both textual and graphical;
- Operation control (proving, batch, sampling)
- Certified gas & liquid calculations
- User defined calculations
- Virtual Flow Computing
- Reporting (Custody transfer)
- Alarm management
- Real-time & historical trending
- Security, audit trail and event log
- Database storage and retrieval
- System and communication redundancy
- Multi-lingual
- Virtual printer 'Flow-Xprint'

1.2. Application examples

eXlerate applications are used in different areas:

- Custody transfer metering
- Allocation metering
- Virtual flow computing
- Terminal automation
- Calibration facilities
- Leak detection

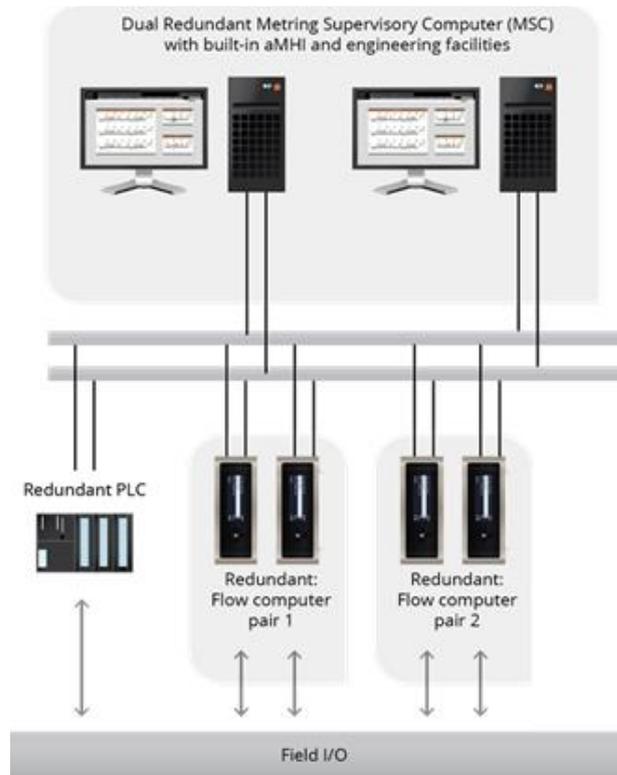


Figure 1 Control systems layers

1.3. Cyber security

This product is designed to be connected to and to communicate information and data via a network interface. It is your sole responsibility to provide and continuously ensure a secure connection between the product and your network or any other network (as the case may be). You shall establish and maintain any appropriate measures (such as but not limited to the installation of firewalls, application of authentication measures, encryption of data, installation of anti-virus programs, etc.) to protect the product, the network, its system and the interface against any kind of security breaches, unauthorized access, interference, intrusion, leakage and/or theft of data or information.

ABB B.V. and its affiliates are not liable for damages and/or losses related to such security breaches, any unauthorized access, interference, intrusion, leakage and/or theft of data or information.

1.4. Manuals

The Spirit^{IT} eXLerate manual set consist out of the following documents:

Installation manual

This manual describes the installation of eXLerate software and all required software on a computer system, as well as how to configure the system and eXLerate options for running Spirit^{IT} eXLerate applications.

Configuration manual

This manual introduces the principles and techniques of real-time application development, as well acts as a reference manual, in which all details of application engineering can be found.

This volume describes many topics, such setting up communication, create HMI displays and reports, utilize calculation worksheets, configuring and structuring your application.

Spirit^{IT} function library

This library describes the functions for flow, gas and liquid calculations used in ABB Spirit^{IT} products.

1.5. Target audience

This manual is written for a variety of readers:

Application developers

System integrators interested in all details required to set-up and develop a complete real-time application with Spirit^{IT} eXLerate.

IT departments

IT experts of companies who are centralized managing software installations on their company systems.

Interested persons

People investigating whether the capabilities and features of Spirit^{IT} eXLerate will satisfy his/her project requirements.

The reader is expected to be acquainted with the basics of HMI / SCADA visualization software packages and to be familiar with Microsoft Excel.

1.6. Document conventions

A book symbol in the text indicates a reference to another section or manual with more details or other relevant information.



A display symbol in the text indicates that the user can find more details on the subject in one of the example applications.



The exclamation mark symbol indicates an important remark made in the manual requiring special attention.



A tool symbol in the text specifies user instructions: the user is assumed to perform some specific action.



Keyboard keys like function keys, navigation keys etc., are presented with the key text enclosed between '<' and '>' characters.

For example, <F1> refers to function key with the text 'F1' imprinted and <Esc> refers to the key with the text 'Esc' imprinted. When a user is assumed to press and release multiple keys simultaneously, those keys are separated by a '-' dash character, e.g. <Ctrl-F1> or <Ctrl-A>.



1.7. Abbreviations

API	Application Programming Interface An interface that allows an application to interact with an application or operating system, in our case, Spirit ^{IT} eXLerate. Most of the Spirit ^{IT} eXLerate API is implemented through Excel worksheet functions.	which a digital signal is superimposed using Frequency Shift Keying at 1200 bps.
ASCII	American Standard Code for Information Interchange. A set of standard numerical values for printable, control, and special characters used by PCs and most other computers. Other commonly used codes for character sets are ANSI, Unicode, and EBCDIC (Extended Binary-Coded Decimal Interchange Code, used by IBM for mainframe computers).	
COM	Component Object Model Standard for distributed objects, an object encapsulation technology that specifies interfaces between component objects within a single application or between applications. It separates the interface from the implementation and provides APIs for dynamically locating objects and for loading and invoking them (see ActiveX and DCOM).	
CPU	Central Processing Unit	
DCE	Distributed Computing Environment Definition from the Open Software Foundation, DCE provides key distributed technologies such as RPC, distributed naming service, time synchronization service, distributed file system and network security.	
DCOM	Distributed Component Object Model Microsoft's protocol that enables software components to communicate directly over a network in a reliable, secure, and efficient manner. DCOM is based on the DCE-RPC specification and works with both Java applets and ActiveX components through its use of the COM. See also ActiveX.	
DCS	Distributed Control System	
DDE	Dynamic Data Exchange A relatively old mechanism for exchanging simple data among processes in MS-Windows.	
DLL	Dynamic Link Library. A file containing a collection of Windows functions designed to perform a specific class of operations. Most DLLs carry the .DLL extension, but some Windows DLLs, such as Gdi32.exe, use the .EXE extension. Functions within DLLs are called (invoked) by applications as necessary to perform the desired operation.	
EIA	Electrical Industries Association	
GUI	Graphical User Interface	
HART	Highway Addressable Remote Transducer. A protocol defined by the HART Communication Foundation to exchange information between process control devices such as transmitters and computers using a two-wire 4-20mA signal on	
HMI	Human Machine Interface. Also referred to as a GUI or MMI. This is a process that displays graphics and allows people to interface with the control system in graphic form. It may contain trends, alarm summaries, pictures, and animations.	
I/O	Input / Output	
IEEE	Institute for Electrical and Electronics Engineers	
ISO	International Standards Organization	
MES	Management Execution System. A level of monitoring of a process control system that is above the PLC and HMI level, where data analysis and integration with other aspects of a company such as accounting and purchasing play a significant role.	
MIC	Machine Identification Code. License code of Spirit ^{IT} eXLerate which uniquely identifies your computer.	
ODBC	Open Data Base Connectivity. A standardized application programmer's interface (API) for databases. It supports Visual Basic, Visual C++, and SQL for Access, Paradox, Text, Excel and many more database standards.	
OEM	Original Equipment Manufacturer	
OLE	Object Linking and Embedding. A protocol specification by which an object, such as a photograph, a spreadsheet, video, sound, etc., can be inserted into and used by an application. Renamed by Microsoft into 'ActiveX'.	
OSI	Open System Interconnection. An ISO standard for worldwide communications that defines a networking framework for implementing protocols in seven layers. Control is passed from one layer to the next, starting at the application layer in one station, proceeding to the bottom layer, over the channel to the next station and back up the hierarchy.	
OPC	OLE for Process Control. A COM interface specification. Applications which implement the OPC interface can inter-operate without the developer needing to control both the server and client development. By following the OPC interface, clients and servers from different manufacturers can communicate and interact successfully. The OPC interface is designed to offer the types of interactions that are typical of process I/O hardware such as PLC, DCS and direct I/O boards. Spirit ^{IT} eXLerate 2016 is OPC DA 2.05 compliant, and ABB is a member of the OPC Foundation.	
P&ID	Piping and Instrumentation Diagram	
PC	Personal Computer	
PLC	Programmable Logic Controller.	

A specialized device used to provide high-speed, low-level control of a process. It is programmed using Ladder Logic, or some form of structured language, so that engineers can program it. PLC hardware may have good redundancy and fail-over capabilities.

RPC	Remote Procedure Call A form of application-to-application communication that hides the intricacies of the network by using an ordinary procedure call mechanism. It is a tightly coupled synchronous process.
RS232	EIA standard for point to point serial communications in computer equipment
RS422	EIA standard for two-wire differential unidirectional multi-drop serial
RS485	EIA standard for two-wire differential bidirectional multi-drop serial communications in computer equipment
RTU	Remote Terminal Unit
SCADA	Supervisory Control and Data Acquisition
SQL	Standard Query Language
SVC	Supervisory Computer
TCP/IP	Transmission Control Protocol/Internet Protocol. Transmission Control Protocol/Internet Protocol. The control mechanism used by programs that want to speak over the Internet. It was established in 1968 to help remote tasks communicate over the original ARPANET.
TTL	Transistor-Transistor Logic
UART	Universal Asynchronous Receiver & Transmitter
URL	Uniform Resource Locator. The global address for documents and resources on the World Wide Web.
VBA	Visual Basic for Applications. The official name is "Visual Basic, Applications Edition." VBA is Microsoft's common application programming (macro) language for Excel, PowerPoint, Visio, Access, Project, Word, and the Visual Basic programming environment.
XLL	Excel Link Library. Special formatted DLL, which is recognized by Excel as extension library. In an XLL, typically worksheet calculations are defined.
XML	Extensible Markup Language. A specification for Web documents that allows developers to create custom tags that enable the definition, transmission, validation and interpretation of data contained therein.

1.8. Terms and definitions

ActiveX	A family of Microsoft object technologies, formerly called OLE, based on the Common Object Model (COM).
Asynchronous	A type of message passing where the sending task does not wait for a reply before continuing processing. If the receiving task cannot take the message immediately, the message often waits on a queue until it can be received.
Client/server	A network architecture in which each computer or process on the network is either a client or a server. Clients rely on servers for resources, such as files, devices, and even processing power. Another type of network architecture is known as a peer-to-peer architecture. Both client/server and peer-to-peer architectures are widely used, and each has unique advantages and disadvantages. Client/server architectures are sometimes called two-tier architectures.
Device driver	A program that sends data to and receives data from the outside world. Typically, a device driver will communicate with a hardware interface card that receives field device messages and maps their content into a region of memory on the card. The device driver then reads this memory and delivers the contents to the program.
Engineering units	Engineering units as used throughout this manual refers in general to the units of a tag, for example 'bar', or '°C', and not to a type of unit, as with 'metric' units, or 'imperial' units.
Ethernet	A LAN protocol developed by Xerox in cooperation with DEC and Intel in 1976. Standard Ethernet supports data transfer rates of 10 Mbps. The Ethernet specification served as the basis for the IEEE 802.3 standard, which specifies physical and lower software layers. A newer version, called 100-Base-T or Fast Ethernet supports data transfer rates of 100 Mbps, while the newest version, Gigabit Ethernet supports rates of 1 gigabit (1000 megabits) per second.
Event	Anything that happens that is significant to a program, such as a mouse click, a change in a data point value, or a command from a user.
Exception	Any condition, such as a hardware interrupt or software error-handler, that changes a program's flow of control.
Peer-to-peer	A type of network in which each workstation has equivalent capabilities and responsibilities. This differs from client/server architectures, in which some computers are dedicated to serving the others. Peer-to-peer networks are generally

simpler, but they usually do not offer the same performance under heavy loads. Peer-to-peer is sometimes shortened to the term P2P.

Polling	A method of updating data in a system, where one task sends a message to a second task on a regular basis, to check if a data point has changed. If so, the change in data is sent to the first task. This method is most effective when there are few data points in the system. Otherwise, exception handling is generally faster.
Process visualization software	<p>A system for monitoring and controlling production processes and managing related data. Typically, such a system is connected to external devices, which are in turn connected to sensors and production machinery.</p> <p>The term 'process visualization software' in this document is generally used for software with which SCADA software, HMI software, or supervisory computer software applications can be built. In this document, although strictly not correct, the terms 'SCADA', 'HMI', 'supervisory', and 'process visualization' are alternately used, and refer to the computer software applications that can be realized with eXlerate, ABB's PC-based supervisory software.</p>
Protocol	An agreed-up format for transmitting data between two devices. In this context, a protocol mostly references to the Data Link Layer in the OSI 7-Layer Communication Model.
Query	In SCADA/HMI terms a message from a computer to a client; in a master/client configuration utilizing the message protocol with the purpose to request for information. Usually, more than 1 data-point is transmitted in a single query.
Real-time	<p>The characteristic of determinism applied to computer hardware and/or software. A real-time process must perform a task in a determined length of time.</p> <p>The phrase "real-time" does not directly relate to how fast the program responds, even though many people believe that real-time means real-fast.</p>
Registry	The Windows Registry is a hierarchical database that stores low-level settings for the Microsoft Windows operating system and for applications that opt to use the registry.
Resource	Any component of a computing machine that can be utilized by software. Examples include: RAM, disk space, CPU time, real-world time, serial devices, network devices, and other hardware, as well as O/S objects such as semaphores, timers, file descriptors, files, etc.
Synchronous	A type of message passing where the sending task waits for a reply before

continuing processing.

Tag	A 'tag' as used within this document refers to a data point existing in the tag database, with several properties, such as assigned I/O address, current value, engineering units, description, alias name, and many others.
Visual Basic	A graphical programming language and development environment created by Microsoft in 1990, and currently used for scripting in applications like Microsoft Office. All macros in Office are created in Visual Basic.
Web Server	A computer that has server software installed on it and is used to deliver web pages to an intranet/Internet.

2. eXLerate applications

Each eXLerate application is contained within a single configuration file. The file extension is “xlrx”. This application file includes:

- Communication settings
- Tag definitions
- Live values (when communication is active)
- User calculations
- Displays
- Animations
- Reports templates

Additional files are created for storing live data:

- Report files
- Trend data
- Alarms & Events (database)
- Event logs
- Retentive data

2.1. Application Shortcuts

The Control Center contains a list of Shortcuts to your applications, as well as other programs. The latter is useful in a production environment where the user does not have access to the Windows.

See the *‘Installation manual’* for details on setting up the eXLerate Control Center and configuring the application shortcuts.

2.2. Open an application

You can open an existing application from the Control Center, either in runtime mode - communication active and only display pages are accessible - or in design mode - for application engineering. See section *‘Application modes’* for more details on the different modes.



Select the Application Shortcut and click either the **[Runtime]** or the **[Design]** button.

Note that you must be logged in with a sufficient access level to be able to click the buttons.



When the shortcut does not exist, you can add new shortcut by right-clicking in the shortcuts list and selecting **[New Shortcut]**.



Another way of opening an application is by double-click the file in Windows Explorer. If the application shortcut doesn’t exist, it will be

added to the control center and file locations will be set based on the file location.

As eXLerate is careful with your work, a temporary working copy is created in the “%TEMP%” directory. when you open an application. This working copy is the file that is opened while the original application file remains untouched to prevent the application getting corrupted on an unintended shut-down.

During the application startup, various startup messages are logged on the Control Center event log. These messages may be used to closely monitor the entire application startup process and are typically used during the application development process.

Examples of such messages:

```
dd/mm/yyyy hh:mm:ss [ xlCenter ] - Starting '...'
in design mode ...
dd/mm/yyyy hh:mm:ss [ eXLerate ] - Initializing
user application data...
dd/mm/yyyy hh:mm:ss [ eXLerate ] - Initializing
displays...
```

When the application is started, the Control Center itself is minimized, and disappears as an icon into the system tray.

2.3. Create new application

You can start from scratch to scratch using the “MyTemplate.xlrx” example application. It allows you to create a new project in a specific resolution.



Open the template application and follow the instructions. The new application is now automatically created.

You may also create your own company specific template and use that when creating new applications.



To create a new application, copy an existing, resembling or template application, rename the file and open it.

2.4. Save an application

Only when you have opened an application in design mode, you can make changes to the application. You can save an application by pressing the **[Save]** button in the menu. At that

moment, first a backup of the previous application file is stored in the “\XLRX\Archive” directory. Then the working copy is saved on the original file location. This way, you can always go back to previous versions. The number of archive files depend on the settings in the Control Center.

You can ‘Save new version’ of the application. This option performs the same actions as the **[Save]** button, and additionally creates and updates the ‘xVersion’ sheet:

- increments the revision number,
- set name of the application engineer,
- set the description for new version,
- set the status of the application,
- set the eXlerate version used
- set the time of the version update.

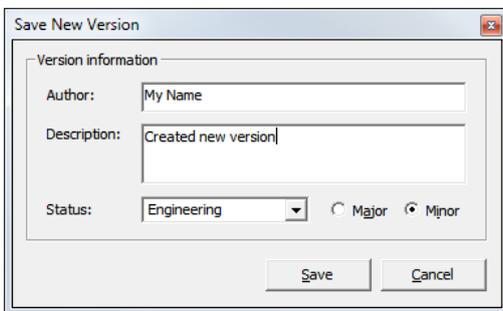


Figure 3 Application Shortcut Properties

2.5. Close an application

You can close a running application, either using the standard menu option from the application file itself, or via the Control Center application shortcut:



In the Control Center, right-click the application shortcut and select **Terminate Application**.

When an application is terminated from the Control Center, eXlerate checks the security level if the user is allowed to close the application. When closed from within the application, this check cannot be performed by the Control Center.

If the application is changed in design mode and not saved yet, you are asked to save or discard the changes.

The progress bar of the Control Center indicates the duration of the shutdown process.

2.6. eXlerate user accounts

Different users will have different access rights. Not everyone is allowed to start, edit or terminate applications, or modify settings. Similar in applications, user levels can be configured to allow or disallow certain run-time actions like alarm acknowledgement, modifying values, etc.

In eXlerate each user has a level number which defines of what the user is allowed to do. The higher the level number, the more the user is allowed to do.

The following standard user and passwords are created when eXlerate Control Center is started the first time.

Table 1 Pre-configured User Accounts

User	Password	Level
guest	guest	10
operator	operator	500
engineer	engineer	1000
administrator	admin	2000



CHANGE THE DEFAULT USER PASSWORDS AFTER INSTALLATION!



Log-in with an **administrator** account, click on the **XL** icon (left top corner) and select **Edit Users** to manage the user accounts.

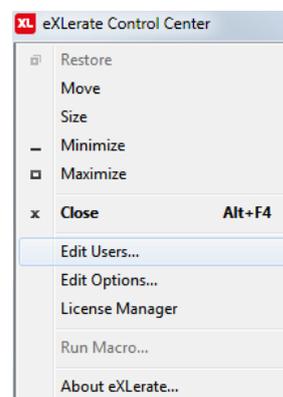


Figure 4 Edit Users

The option is disabled when you have insufficient rights.

You get a dialog with the user accounts and you can add, modify and delete users.

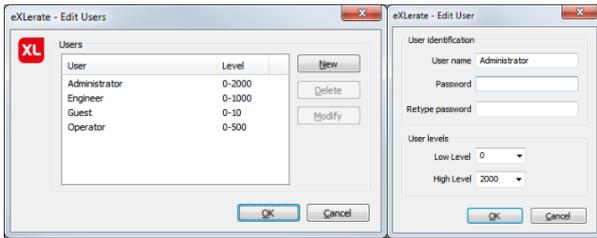


Figure 5 User Account Configuration

For each user set a name, a password, and the low and high levels.

The two levels are used to extend the flexibility. Access is granted if the defined level for an action is between the low and high level of the user.

If this all sounds complex, the “Low Level” setting is typically set to zero, allowing a user to perform all the actions lower-level users can do.

The users are applicable on the local computer for the eXlerate system. Different computers can have different users. The security in eXlerate applications is based on the user level (number); running an application on another computer will use the users as defined on that computer.

3. Application basics

eXlerate is integrated with Microsoft Excel. It allows to use all the available Excel functionality and it adds features to Excel in various ways. The more generally interested reader is expected to be commonly acquainted with Excel. An application developer is assumed to have a thorough understanding of at least the following aspects of Excel:

- Worksheet/workbook organization
- Cell references
- Cell formatting
- Cell formulas
- Named ranges, tables
- Array formulas
- Shapes and pictures
- Macro recording and playback
- Visual basic for applications

With eXlerate extra functionality is added to the standard excel functions:

- Ribbon with Wizards
- Real-time communication
- Buttons for navigation and control
- Animation of shapes & objects
- Trending of data
- Alarm management
- Database
- List views
- Additional worksheet functions
- Additional VBA functions

The most visible part while developing an application is the eXlerate toolbar (ribbon).

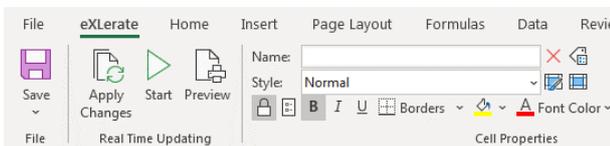


Figure 6 eXlerate toolbar ribbon

3.1. Application modes

Developing an eXlerate application happens in phases: you configure your application, you test if it works, and eventually it is put in production. To facilitate every step, eXlerate can operate in four different modes.

Table 2 eXlerate modes

Mode	Communication	Sheets protection
Design	Inactive	Unprotected
Preview	Inactive	Protected
Verify	Active	Unprotected
Runtime	Active	Protected

3.1.1. Design mode

This mode is to develop the application. Worksheets are not protected, and you don't have real-time updates from your connected devices.

3.1.2. Preview mode

In preview mode, you see the application as your user would see it, i.e. only the display pages. Sheets are protected against making changes, but real-time communications are disabled.

3.1.3. Verify mode

Verify mode is for debugging your application. Real-time communication with devices is enabled but sheets are unprotected, so you still see all the worksheets and can debug the configuration, communications, and calculations.



Warning: do not edit cells, edit VBA code or save your application in verify mode.

3.1.4. Runtime mode

This mode is for normal operations. This is what end-users will use on live systems: real-time communications are enabled, only the display screens are visible, and the sheets are protected against making design changes.

3.1.5. Switching modes

You can start the application directly in Runtime mode or Design mode from the Control Center. Switching between modes is done by clicking the appropriate buttons.



To go from Design mode to Preview mode, press the **[Preview]** button in the eXlerate ribbon or press <Ctrl-N>.



To go from Preview mode back to Design mode, press <Esc> followed by **[Design Mode]** button that appears.

-  To go from Design mode to Verify mode, press the **[Start]** button in the eXlerate ribbon or press <Ctrl-T>
-  From Verify mode you can go to Runtime mode. Click the **[Runtime]** button in the eXlerate ribbon or press <Ctrl-N>.
-  Press <Esc> followed by **[Verify Mode]** button to go from Runtime mode to Verify mode.
-  To exit Verify Mode and go to Design mode, press the **[Stop]** button in the eXlerate ribbon or press <Ctrl-O>.

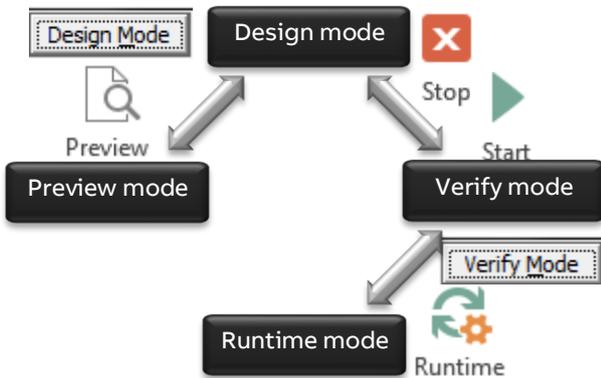


Figure 7 Switching modes

3.2. Excel basics

While developing an application you make use of Excel’s features. While this document is not intended to describe how Excel works, we do present some of the basic functionality in this.

3.2.1. Worksheets

A worksheet is a collection of cells where you keep and manipulate the data. Each Excel workbook can contain multiple worksheets. Within an eXlerate applications, multiple worksheets are available, all with their specific functionality like configuration settings, calculations, displays, reports.

3.2.2. Cell data

In Excel cells can contain (numerical) data. This data can be presented in different formats (styles), like numbers with different decimal places, date, time, text, etc.

3.2.3. Shapes, pictures and charts

With Excel you can put a variety of standard shapes, pictures, and charts on sheets. With

these, you can create drawings and graphical presentation on sheets. With eXlerate it is possible to animate these objects based on live values, see section ‘Animations’.



Figure 8 Insert Shapes, Pictures and Charts

Shapes can be grouped together to form new shapes. Limit the grouping of shapes to one (1) level to allow animations of the sub-shapes of grouped objects.

 **Don't use cell references** to dynamically change the text on shapes as it causes memory leaks with excel.

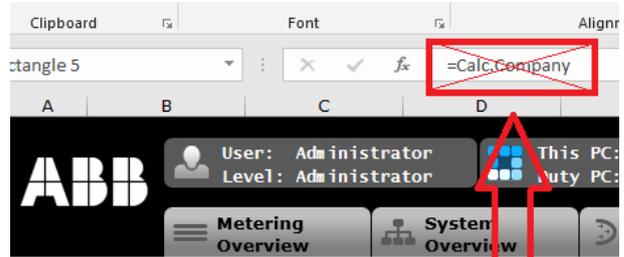


Figure 9 Don't use cell references on shapes

 **Don't use 3D effects** as it is know that it can cause memory leaks with excel.

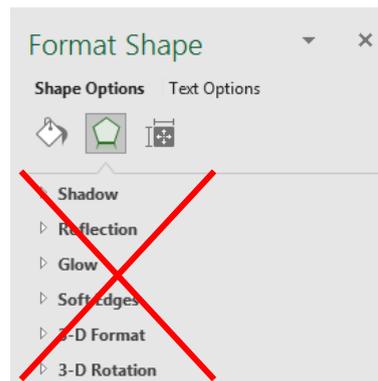


Figure 10 Don't use cell 3D shapes

3.2.4. Naming cells and shapes

Giving logical names to cells and objects (shapes) makes it easier in an application to understand the meaning is of a cell value in your application and to use eXlerate animations for shapes.



To give a name to a cell or a shape, you enter the name directly the **Name** input on the eXlerate ribbon.

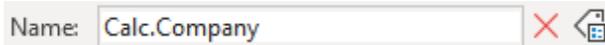


Figure 11 Name definition

For cell names, you can also use the **Name Manager**, which shows all the defined names, checks the referred cells, and check for errors.



During application development regularly check for errors or invalid references:

- Check there are no references to other workbooks, like
`= 'File Path\[Application.xlsx]Sheet'!A15`
- Check there are no “Names with Errors”
- Limit the use of “Names Scoped to Worksheet”
 Same names with different worksheets scopes may result in unexpected values as it is not clear which scope is being used.
 Note that it is used by excel Page Setup for print setting per sheet.

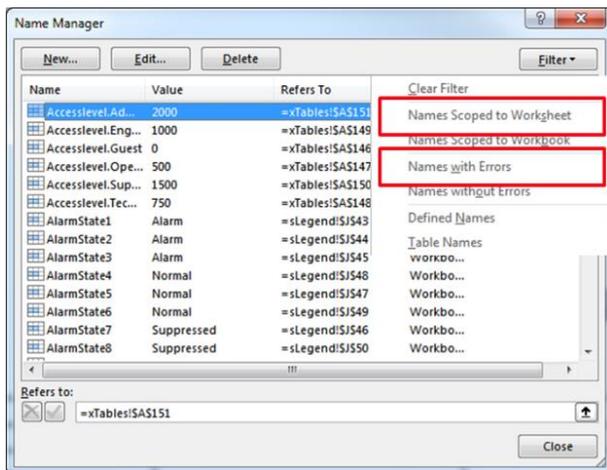


Figure 12 Name Manager

3.2.5. Formulas

You can use formulas in cells to create your own calculations. A formula always starts with an equal sign (=). You can use logical operators, Excel functions and additional eXlerate functions.



The automatic calculation of formulas is disabled in eXlerate to prevent a torrent of calculations. In Runtime mode eXlerate evaluates formulas once a second. When you want to see the result of a change you made in Design mode, press the **<F9>** key.

Reference to other cells

You can refer to data in other cells and on other sheets to be used within formulas

- =G10
 Relative references to a cell address on the same sheet. The reference changes when the cell is copied.
- =\$G\$10
 Absolute references to a cell address on the same sheet. The references don't change when the cell is copied.
- =xTagDB!G10
 Relative references to a cell address on another sheet. The reference changes when the cell is copied.
- =xTagDB!\$G\$10
 Absolute references to a cell address on another sheet. The references don't change when the cell is copied.
- =Calc.Stn_GVR_CUR
 References to a named cell. The references don't change when the cell is copied.

Formulas can have other cells as input. This can be any of the references to cells as described above. For example, a cell contains the formula which adds the values of cells **A1** and **A2**:

=SUM(A1,A2)

Rather than use cell addresses, you can also use the cell names, like:

=SUM(xSTR1_GVR.Value, xSTR2_GVR.Value)



Using named cells makes it better to understand what the formula is calculating.

Array formulas

Most formulas have a single result value. Some formulas however produce an array of results. Excel has two different methods of handling arrays, depending on excel version. When you use a formula in dynamic array aware Excel, it determines if the formula has the potential to return multiple values. Excel formulas that return a set of values, return these values automatically into neighboring cells.

Figure 13 Excel dynamic array

To enter an array formula in older versions, select as many cells (rows and columns) as necessary for the results. Then enter the formula and press <Ctrl-Shift-Enter>. Excel will automatically enclose the formula in curly brackets ({}).

Figure 14 Excel Control-Shift-Enter formula

Volatile functions

A volatile function causes a cell is always recalculated even if the input values have not changed. This results in a torrent of calculations, decreasing the performance of the application.

⚠ **Don't use volatile functions** in your application, but their alternatives:

- Now() -> exNow()
- Today() -> exNow()
- Offset() -> Index()
- Indirect() -> (No alternative)
- Rand() -> eXlerate simulation
- Cell() -> exCellProperties()
- Info() -> (No alternative)

Circular references

A circular reference occurs when a formula, direct or indirect, refers to its own cell. It then- uses its result as input. This causes iterative calculations, leading to erratic behavior and decrease of the application performance.

⚠ **Perform error checking** to check and solve circular references regularly during design.

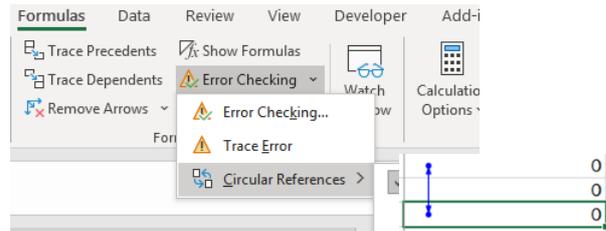


Figure 15 Check for Circular References

3.2.6. Visual Basic for Applications

Excel has a built-in scripting language called VBA. You can use this VBA to create your own forms (pop-ups) and functions. To start the VBA editor, press <Alt-F11> or menu **eXlerate | Visual Basic**.

eXlerate adds many functions you can call from VBA. You can find a help file with all available functions by clicking **Help | Function Reference**. These functions allow you to manipulate eXlerate specific items such as alarms, trends, etc.

In an eXlerate application, there are some VBA modules present which are automatically generated, like the module “modButtons”.

⚠ You should not change these modules, for your changes will get overwritten when running the eXlerate wizards.

3.3. eXlerate worksheets

An eXlerate application consist of specific formatted worksheets in a single workbook. Several sheets are interpreted by eXlerate in a special way to produce user interface displays, reports, and perform calculations at runtime.

The data and the calculations on the sheets are interconnected, which may seem overwhelming when first opening an eXlerate application. However, most applications follow the same pattern and set-up.

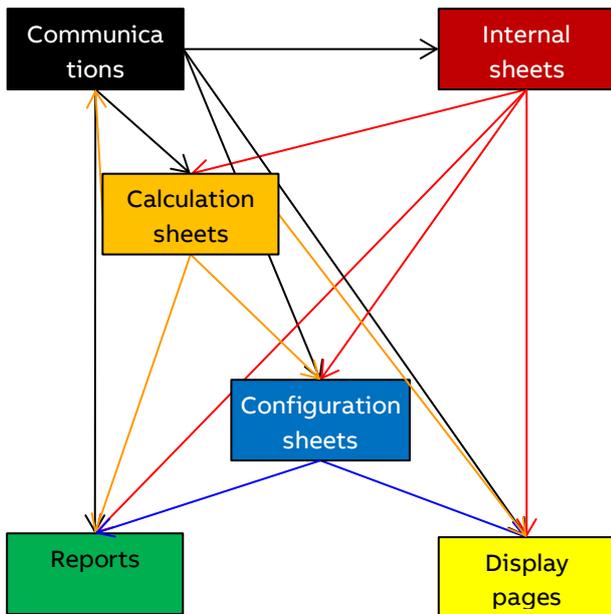


Figure 16 eXlerate sheets relations

3.3.1. ■ Communication sheets

The backbone of an application are tag database and the communication set-up.

The **xComms** sheet contains the configuration for communication with external devices, e.g. the COM/IP ports, baud rates etc., and the definitions for polling / writing the data blocks. It also contains debugging facilities for communications.

The sheet **xTagDB** is the tag database. It lists all tags that are communicated with external devices with their properties and the (live) values.

3.3.2. ■ Internal sheets

The **Tag & Object Wizard** will automatically generate the internal sheets based on the tags and defined properties. These internal sheets have named cells with values that will be updated when the communication is running.

3.3.3. ■ Calculation sheet

For calculating values, you use a calculation sheet. You can use all Excel and additional eXlerate functions. You can have multiple calculation sheets. Their names must start with 'xCalc'.

3.3.4. ■ Configuration sheets

An eXlerate application contains additional configuration sheets that influence the behavior or the looks of displays and reports.

The **xTables** sheet contains the configuration tables, like which sheets are displays (available in runtime), which sheets are report templates, interval & periods for event-based calculations like averaging and automatic report generation, definitions for buttons, and more. It contains the following configuration tables:

- Alarm group table
- Interval table
- Report table
- Style table
- Color table
- User table
- Worksheet table
- Button table

The **xAnimations** sheet contains the definitions to animate shapes. You can have tag values and calculated values affecting the colors, position, size or visibility of a shape on your display sheets.

The **xEditing** sheet contains the definitions to allow users to change values (settings) when the application is in runtime. The values can be used internally and be sent to connected devices.

3.3.5. ■ Displays

Display sheets are the runtime user interface presenting the status and values to the user. It can contain references to live and calculated values, shapes and object displaying the field status, buttons for navigation and control.

3.3.6. ■ Reports

Report sheets serve as a template for generation of reports. It contains references to live and calculated values. When a report is generated, a snapshot of these values is taken and stored in separate report files on disk.

3.3.7. ■ Safety precautions

To prevent regular users from debugging and going into the VBA source code during runtime, avoid using Excel's built-in **Worksheet.Change** event. Also consider calling **exRunCancelKey()** before any other code within a custom subroutine as that will prevent debugging VBA code executed below that call for that subroutine for unauthorized users.

Note that the usage of this command won't work on earlier versions of eXlerate and that if you

want backwards compatibility with older versions, you should look into the implementation of `exRunCancelKey()` that is generated within `modButtons`.

3.4. eXlerate wizards

The eXlerate menu provides “wizards” for automation and enhanced configuration of the application. Wizards perform automatic naming, generate internal sheets and calculations, add alarm lists navigation buttons, and more.

This section describes the general functionality of the wizards. How to set-up your application and work with these wizards is described in the following sections in this manual

The wizard reports the progress and found errors into the output window. These error messages are clickable. Double-click on the error will select its location in the application.

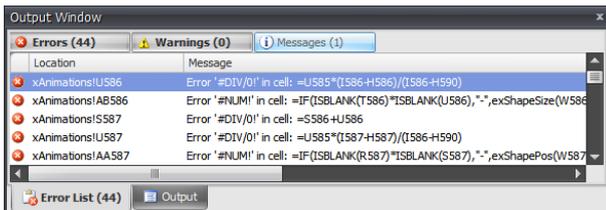


Figure 17 Wizards output window



3.4.1. Tag & Object wizard

The ‘Tag & Object wizard’ performs the following actions:

- Create cell names.
 - Named cells makes an application easier to understand - see *‘Naming cells and shapes’*.
 - Names are created for cells with:
 - Live values
 - Alarm settings
 - Alarm statuses
 - Average values
 - Latched values
- Remove invalid names
- Create the actual alarm list
- Create trend tags
- Update the communication configuration
- Check the entire application for errors



Run the **Tag & Object wizard** from the ribbon or by pressing <Ctrl-W> after making changes to the communication sheets.

After running this wizard, several cell reference names are available to you. These names are based on the name you specified and prefixed by a lowercase ‘x’. If you define a tag with the name **‘MyTag’**, the reference names **‘xMyTag.xxx’** will be created, with **‘xxx’** being the specific property/field of the tag. See section *‘Tag Database’* on the field and cell names.

3.4.2. Calculation wizard



The ‘Calculation Wizard’ generates the names for the cells on calculation sheets.

A worksheet is a calculation sheet when the name starts with ‘xCalc’, and where on the third row a header line is defined containing fields: ‘Group’, ‘CalcTag’, ‘Description’ etc.

The created cell names are the concatenation of the name defined in the *‘CalcTag’* field and the field header name when the field name starts with an underscore, except for the *_Value* field, for which the name will only be the *‘CalcTag’*.

For example:

CalcTag: *Calc.Str1_MOV1:*

Field name: *_Value*

Cell name: *Calc.Str1_MOV1*

CalcTag: *Calc.Str1_MOV1*

Field name: *_Position*

Cell name: *Calc.Str1_MOV1_Position*

CalcTag: *Calc.Str1_MOV1*

Field name: *Text*

Cell name: *(no name)*



Run the **Calculation wizard** from the ribbon after making changes to a calculation sheet.

The calculation wizard can only be invoked when the selected worksheet is a calculation worksheet.

3.4.3. Button wizard



The ‘Button wizard’ creates the buttons actions and functionality.

The wizard will parse the configuration for the buttons - see section *‘Button table’*. It will:

- Create / update VBA module *‘modButtons’* with the procedures for display navigation
- Apply security checks on these procedures

- Update text on buttons
- Assign procedures to the buttons
- Assign procedures to keyboard keys



Don't make changes to this VBA module '*modButtons*' as these changes will be lost when running the wizard.



Run the **Button wizard** when you made changes to the Button table.

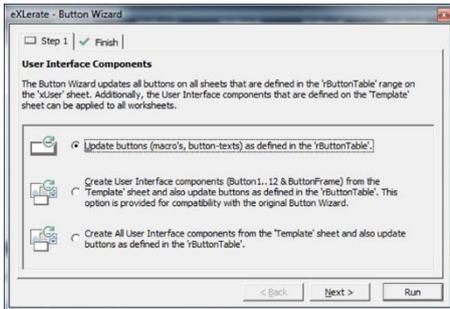


Figure 18 Button Wizard

When you start the Button Wizard, it presents you with the following options:

- Update buttons
Only update the text on the buttons and the assigned macros.
- Create User Interface components
(*Only for backwards compatibility*)
Creates the objects 'button1'-'button12' position/layout on the display sheets according the '*Template*' sheet and updates the text and assigned macros.
- Create All User Interface components
Creates all objects as defined on the '*Template*' sheet on the display sheets and updates the text and assigned macros.

You can test the display navigation by switching to preview mode.

3.4.4. Color wizard



The animation colors are configuration in the '*Color table*'. The 'Color wizard' updates these colors for use in runtime.



Run the **Color wizard** when you have made changes to the Color table.

3.4.5. Language wizard



Multi-lingual applications require the '*xLanguage*' sheet with the translations for

product and application related texts.

The 'Language wizard' initially creates the multi-lingual configuration sheet '*xLanguage*' with the product related text strings. A user can add languages and application related text string that require translations to this sheet.



Run the **Language wizard** when you are implementing multiple languages, after making changes to the language sheet.

Existing translated texts will always remain unaffected by the Language wizard.

3.4.6. eXlerate engineering tools

The eXlerate ribbon provides additional utilities useful during the developing of an application. The usage of the tools is explained in more details in the subsequent sections.

- **Shape property tool**
This tool is deprecated as the Excel's default *Selection Pane* <Alt-F10> and *Format shape* <Ctrl-1> tools provide the same functionality.
- **Name definition tool**
This tool is deprecated as the default Excel's *Name manager* <Ctrl-F3> provides the same functionality.
- **Color palette tool**
This tool is deprecated as the information provided is available in the *Color table*.
- **Alarm tree tool**
This tool is deprecated as the information provided is available in the *Alarm groups table*.
- **Generate Report**
This option allows you to quickly generate a report for engineering purposes.
- **Browse OPC Servers**
This tool allows you to browse local and remote OPC servers, search item names and paste items into the application.
- **Communications options**
This shows the diagnostic and logging options for the communication protocols.
- **Mark/Unmark unprotected cells**
This tool is deprecated as marking/unmarking protected cells can result in losing cell formatting.
- **Remove external links**
Removes all external links in an application workbook and references are reverted to the current application.

- **Reset historical values**
This option resets the calculated averages and latched values of the current application. It can be useful for example restart calculating averages and latching after an application test.
- **Recalculate application**
This option triggers the application recalculation. The system flag: 'xAutoRecalc', which is used to force recalculate expressions in Excel on a system restart is updated as well. and shape animations are updated.
- **Import sheets**
This option allows you to easily import sheets into your application. The tool automatically removes any external links and provides the ability to replace content while importing.
- **Advanced replace**
Advanced Replace allows you to find & replace content not only on the sheet, but also in names, and objects. Special 'prefix' and 'postfix' options allow you to replace only the content that you are interested in.

4. Data communication

Real-time data communications in eXlerate enables the exchange of information with one or more external devices, such as flow computers, and Process Logic Controller



Figure 19 Data communication

4.1. Communication model

To connect devices, specific hardware such as Ethernet interface cards or serial ports, are used. Also, the data exchange needs to take place in an agreed format. A communication protocol in eXlerate is the combination of the physical connection and the data message type is

A protocol is usually query based. A query defines the data items, the direction (from client to server or vice-versa), and how this transfer takes place: periodically, event-based, manually, or a combination of these methods. Queries only ask for data (a read-only query), only write data (a write-only query) or have combined read/write commands. A 'query' in this manual is also referred to as a 'poll-block', 'message', or 'frame'.

Usually, more than 1 data point is transmitted in a single query. In eXlerate, a single tag value corresponds to a single data point.

Communication protocols in eXlerate have a multiple query-based structure, i.e. each of the n protocols in a project has m data queries in which j data-points (tag values) are enclosed.



Figure 20 Data communication

In eXlerate, the protocols run asynchronously in independent programs execution threads, and process their associated queries sequentially.

Many manufacturers have developed many different protocols and variations, each with certain advantages and drawbacks.

In eXlerate, the ActiveX control 'xlConnect', is responsible for real-time data communications. It is the intermediate between the device and the application.

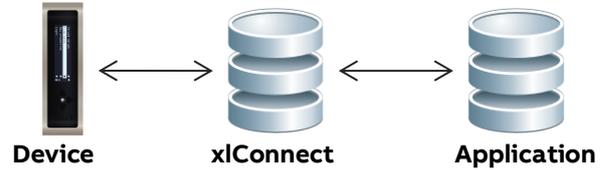


Figure 21 Data communication flow

The following communication protocols are available in eXlerate (depending on license):

- Flow-X Client
- Modbus TCP Client / Server
- Modbus Serial Master / Slave
- OPC Client / Server

4.2. Set-up communications

Real-time data communications in an eXlerate project is configured in two worksheets:

xComms

This worksheet contains the protocol table with the definitions for the (physical) connections, the query table with the message queries for each protocol and the ActiveX control 'xlConnect'.

xTagDB

The tag database worksheet contains the definitions for all tags communicated with external devices, including the properties needed for data communications, and the live values.

4.2.1. Protocol table

The Protocol Table is list that contains the definitions of the protocols to communicate with the devices. Multiple protocols are available, all running in parallel. For each protocol, a thread is created in Windows. For redundancy, a device can have multiple protocols defined.

Protocol Table										
ID	Protocol	Type	Options	MoreOptions	Device	iceOptions	MoreOptions	Tag	Description	Status
1	FlowXClient				192.168.1.100		200,90,90	FXM1	FlowX link 1	0
2	FlowXClient				192.168.2.100		200,90,90	FXM1	FlowX link 2	0
3	ModbusClient	RTU			192.168.1.100		502,50,50,100	FXM1	Modbus link 1	0
4	OpccClient	OPC 2.0	eXlerate		1			OPC	OPC link	0

Figure 22 Protocol table

For each specific protocol in eXlerate, an example application is available. These



applications contain the available options, settings, and other parameters. Check the example applications for additional information of the protocols that you need in your application.

The following Protocol Table columns are relevant:

- **ID**
Unique, incrementing index number internally identifying the protocol. This ID will be used in the Query table.
- **Protocol**
The key name of the protocol type, for example: 'FlowXClient', or 'ModbusMaster'.
- **Type**
(optional) Message type for the protocol, such as 'RTU', or 'ASCII' in case of Modbus, or 'OPC 1.0' or 'OPC 2.0' in case of OPC Client.
- **Options / MoreOptions**
At these fields, specific protocol dependent options may be entered.
- **Device**
Specifies the device to use, i.e. a serial port configuration (like "COM56:9600,n,8,1"), or a in case of an Ethernet device the TCP/IP address (like "192.168.0.10"). It can be set to 'Sim' to simulate the device so the protocol and pertaining data is simulated.
- **DeviceOptions**
Device connection options, like IP port number or serial RTS signal set-up
- **DeviceMoreOptions**
Connection fail and reconnect timing settings
- **ModemInit**
In case a modem is connected, the initialization string sent to the modem prior to calling.
- **DialCommand**
Dial command for the modem, such as: "ATDT +31402961234".
- **MOptions**
Modem dialing timers and reconnect counters
- **UserName**
Username for connecting (Flow-X protocol)
- **Password**
Password for connecting (Flow-X protocol)
- **Tag**
(optional) tag name of the device. May be left empty as it is not internally used.
- **Description**
(optional) A description of the device. May be left empty as it is not internally used.

- **Status**
This value represents the communication status of the protocol. Value of zero (0) means no communication, another value status depends on the protocol type.

4.2.2. Query Table

The Query Table defines poll blocks (message queries) of the data to read from/write to the devices. Each row in the Query Table is linked to one protocol (refers to the ID the Protocol Table), each protocol can have 1 or more queries. The data of a query is related to the tags in the xTagDB.

Query Table												
ID	Protocol	Device	Interval	Timeout	Retries	Sleep	Row	Col	Worksheet	Option	MoreOptions	Status
1	1	1	20	5	2	100	16	7	xTagDB			1
2	2	1	20	5	2	100	16	7	xTagDB			1
3	3	1	20	5	2	100	126	7	xTagDB			1
4	3	1	20	5	2	100	148	7	xTagDB			1
5	3	1	20	5	2	100	150	7	xTagDB			1
6	3	1	20	5	2	100	154	7	xTagDB			1
7	3	1	20	5	2	100	164	7	xTagDB			1
8	3	1	20	5	2	100	174	7	xTagDB			1
9	3	1	20	5	2	100	184	7	xTagDB			1
10	3	1	20	5	2	100	189	7	xTagDB			1
11	3	1	20	5	2	100	193	7	xTagDB			1
12	3	1	20	5	2	100	195	7	xTagDB			1

Figure 23 Query table

The important columns in this table are:

- **ID**
Unique, incrementing number identifying the query. This ID will be used in the xTagDB.
- **Protocol**
Link to the Protocol Table 'ID'. It designates the query to a specific protocol.
- **Device**
Device/Slave ID used in each message (for multi-drop protocols), e.g. Modbus, this is a number; for OPC, this is a group name
- **Interval**
The time between two consecutive read- or write-polls, and is entered in 0.1 sec units, e.g. 30 = 3.0 second intervals. The relative start time of a query may be also defined, e.g. when the user specifies: '20:15', there will be a 2.0 second interval update for this query, which starts after 1.5 seconds elapsed.
- **Timeout**
The maximum time, in 0.1 sec units, to wait for a response. When no response is received within this time, a retry of the query is sent.
- **Retries**
The number of retries in case of a timeout is specified with this parameter. When elapsed, the device is set to sleep.
- **SleepTime**
This parameter specifies the sleep time, in 0.1 sec units. The associated device is put to sleep

when all retries have been elapsed and still no response is received at a message query. A sleep time is used to let other devices, which share the same protocol hardware, prevail communications over a failing device. When the sleep time has elapsed, communications are resumed.

– **Row**

The row number of the tag database with the first item value for this query. Usually, the Excel MATCH() function is used to automate the lookup of these values.

– **Col**

The column number of the tag database that contains the values for this query. Usually, the Excel COLUMN() function is used to automate the lookup of these values.

– **Worksheet**

Specifies the worksheet name with the tag database – always set to 'xTagDB'.

– **Options**

Advanced read and write options. These options may be combined, by adding individual settings.

- **xBlockWrites**
Write all items in a query at once rather than individual items when either value belonging to the query has changed. By default, items may be individually updated.
- **xNewDataOnly**
Send only new data to Excel. When received data is equal to the data already available, no update takes place. This option prevents unnecessary calculation updates. By default, every read query causes an associated update in Excel.
- **xTransparentRead**
Allow reading data while write updates are currently pending. By default, read polls are postponed during a write update.
- **xForcedWrites**
Allow pending write commands to be executed after a device has gone back on-line. When a device has gone off-line, write queries cannot be sent to a device. This option allows for automatic update of the data once the device is online again. By default, no automatic writing takes place.
- **xNoReadOnce**
Disable an initial read. By default, write-only queries are updated initially, there is a read command defined for the query.

- **xItemUpdates**
Items in this query are updated in Excel individually rather than group-wise, a query at a time. This option is available for the OPC client protocol.
- **xNoSleepAll**
Prevents other queries from going to sleep when one query in the protocol fails. This option is available for Modbus protocols.
- **xWriteOnly**
Items only will be written and are never updated in Excel. This option is available for the OPC client protocols.
- **xWriteAll**
Writes the whole query-block, when at least one item in the query was changed.

– **MoreOptions**

Enable query logging of all the received data.

– **Status**

When the query is on-line, eXlerate writes a "0" (zero) into this cell. If a query goes off-line, a "1" (one) is stored in this cell.

Additional fields are protocol dependent. Check the working sample workbooks for an example of the protocol you require.

For **Flow-X** protocols, the Query table contains no protocol specific columns.

For **Modbus** protocols, the Query table contains the following columns:

- **14: Type**
Poll-block type:
'1': coils; '2': registers; '5': OMNI report.
- **15: Address**
Start address of the query.
- **16: Length**
Number of registers/coils for this query.
- **17: Size**
Number of bits in a register (1, 16, 32, 64, 128).
- **19: SW**
Supported Modbus single write command:
'5': coils; '6': holding registers.
- **20: MW**
Supported Modbus multiple write command:
'15': coils; '16': holding registers.
- **21: MR**
Supported Modbus multiple read command:
'1': coils; '2': discretes; '3': holding registers; '4': input registers; '65': OMNI text buffer.

For **OPC** client protocols, the Query table contains the following columns:

– 14: Deadband

Reject values that do not change more than a certain percentage from the previous value recorded.

4.2.3. Advanced Read mode

Normally, devices are polled periodically for data according the interval fields defined at the Query Table. With the VBA function *exSetReadMode()* you can alter this default behavior and specify how to read the values from external devices.

- **exCommsReadModelInterval**
Update the query at its defined interval. This is the default mode. Use this to restore the default original behavior.
- **exCommsReadModeDisabled**
Suspend reading polls of the specified queries.
- **exCommsReadModeTrigger**
Read the specified queries, but only if not scanned periodically.
- **exCommsReadModeTriggerAll**
Read the specified queries, even if periodically scanned.

4.3. Update device values

With eXlerate it is possible to write values to a connected device in three different ways:

- Display Editing functions
Operator entries from displays, e.g. override values, alarm limits, settings
- Tag Database update functions
Live/calculated data available in application, e.g. selected gas composition, updates to DCS
- Visual Basic functions
Control signals from VBA procedures, pop-up forms, e.g. valve commands, proving commands

4.3.1. Communication settings

To be able to send values to a device, make sure that eXlerate can update the device value over the communication link. For Modbus Master/Client protocols the single write (SW) and/or multiple write (MW) function codes should be enabled in the query table (and in the connected device). For Modbus Slave/Server protocols the multiple read (MR) function codes should be enabled in the query table (and in the connected device).

4.3.2. Display editing

When you want a user input from a display to be written to a device, you can use the editing functions as defined in section *'Editing values'* above. Unlock the cell(s) on the display to allow user input in runtime, and configure the Editing table for these cell(s) with the target type set to **"xTargetComm"** and the Target to the tag value in the tag database (like **"xTagName.Value"**).

When the user enters a new value in the cell in runtime, the value is sent to the device.

4.3.3. Tag Database updates

When you want to send (live or calculated) values to a connected device, you can use the update functions in the Tag Database. Define the reference to the value / calculation in the **WriteValue** column and add in the **Update** column an eXlerate update function to trigger the data update to the external device:

- **exUpdateEx()** for numerical values
- **exUpdateStrEx()** for string values
- **exUpdateVarEx()** for variant values

These functions require the following arguments

- **Query ID**
Query to use for communication.
Refer to the correct *Query* column.
- **Tag ID**
Internal ID number of the tag
Refer to the *ID* column
- **Value**
Value to write to the device
Refer to the *WriteValue* column
- **Update mode**
Mode when the update should take place
 - xUpdateNever Don't update
 - xUpdateAlways Always when triggered
 - xUpdateConditionally Update as set by VBA function **exSetUpdateMode()**.
- **Trigger**
Optional trigger to update.
When omitted, the update is triggered when the write value changes.
When set, the update is triggered when either the value or the trigger changes.

Example

To write a new value for the tag with on row 73, the function looks as follows:

=exUpdateEx (L73, B73, H73, xUpdateAlways)

With L73 Reference to cell with the Query ID
 B73 Reference to cell with the Tag ID
 H73 Reference to cell with the value
 xUpdateAlways Always update

4.3.4. Visual Basic updates

You can create your own user input and procedure with VBA and user forms. When you want to update values from your VBA code to a connected device, you can use the same three functions as the worksheet update functions:

- **exUpdateEx()** for numerical values
- **exUpdateStrEx()** for string values
- **exUpdateVarEx()** for variant values

You specify the Query ID, Tag ID, the write value, and the update mode as arguments for these functions.

In VBA you have an additional function:

- **exUpdateForce()** for numerical values

This function triggers the value is sent to the device, even if the value has not changed.

This function requires as arguments the tag name like **xTag.Value** rather than the Query ID and Tag ID, and the value to write. As optional argument you can specify if the value in the tag database is to be updated as well, avoiding inconsistencies with the tag database.

4.3.5. Advanced Update mode

Within VBA you can specify how the values are updated to external devices with the function **exSetUpdateMode()**.

The function defines how the *'exUpdateEx'*, *'exUpdateStrEx'*, *'exUpdateVarEx'* functions update the values for the corresponding protocols and queries when their mode parameter is set to *'exCommsUpdateConditionally'*. Other modes are no effected.

The Mode supports the constants as defined in the exCommsUpdateSetMode enumeration.

- **exCommsSetModeDisableAll** (4)
No values are written, although the values are internally updated.
- **exCommsSetModeNewItemOnly** (5)
From this moment on, values are written.
Values that have been updated previously are

discarded, e.g. only newly updated values will be written to the device.

- **exCommsSetModePendingItems** (6)
Write all values that were internally updated while mode was disabled and remain active.
- **exCommsSetModeFlushActivate** (7)
Write all once initialized values and remain active.
- **exCommsSetModeTriggerOnce** (8)
Write all values that were internally updated while mode was disabled. After the update remain inactive and wait for another update trigger.

4.4. xlConnect

In eXlerate, the component **xlConnect** manages all data-communications. It is implemented as an ActiveX control on the 'xComms' worksheet. It is the "intermediate" between external devices and the application. It uses the configuration as defined in the application Protocol table and the Query Table to connect to and communicate with external devices.



Run the **Tag & Object Wizard** after making changes to the Protocol Table or the Query Table to load these changes into **xlConnect**.

Communications are controller by the [▶ Start] and [[X] Stop] buttons on the ribbon. You will see the icons in the top-right of **xlConnect** change depending on the communication status.

The **xlConnect** component allows you log and debug the communication with the devices. The main window contains a couple of buttons, icons and a message area for logging and debugging.

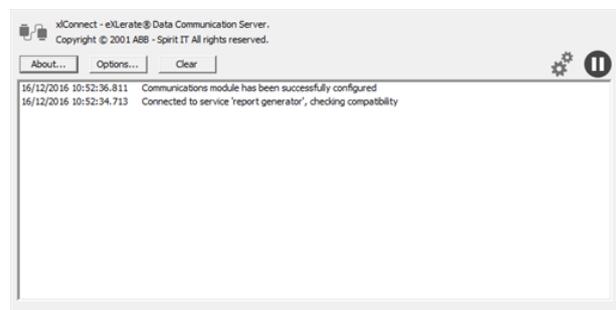


Figure 24 eXlerate data communication layout

- **[About...]**
Button to show version and information about the product.
- **[Options...]**
Button to set logging and debugging options

for the individual protocols and queries and for cyclic interval events. See details below.

- **[Clear]**
Button to clear the message area.
- **[⚙️]**
Icon showing if cyclic interval events and period updates are currently active and triggered in excel.
 - ⚙️ light grey and standing still: not active.
 - ⚙️ dark grey and in motion: active.
- **[🔇]**
Icon presenting the current status of the xLConnect ActiveX control to the user.
 - ✅ The communications are correctly configured. Real-time data update is active.
 - 🔇 Communications are stopped, because the user has clicked on this icon, or stopped from the menu.
 - ⏸ Communications are programmatically paused.
 - ⚠ The communication configuration contains warnings.
 - 🔧 The configuration correctly set up programmatically (via VBA).
 - 📄 The configuration is not defined yet for xLConnect.
 - ❌ The communication configuration contains errors - details are logged in the system event logger
Correct the problem prior to starting the communication.
- **Text messages**
Local message window, used as a data scope logger and a fault/warning message window for data communications. See details below on how to enable logging of message.

Besides protocol management, it also takes care of the cyclic interval processing. Cyclic interval processing is further discussed in section *'Interval periods and events'*.

4.4.1. Logging and debugging

The xLConnect allows to debug and log events and communications. Press the **[Options...]** button to set it up.

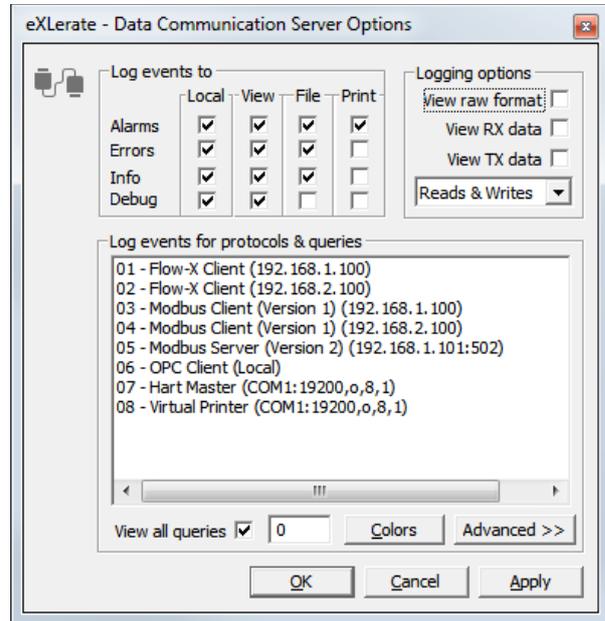


Figure 25 Event and communication logging

The top left section is the set-up for event logging. There are various message types: 'Alarms', 'Errors', 'Info', and 'Debug'. You can select the log devices for the events:

- Local xLConnect local window
- View Control Center message window
- File Log-file on disk
- Print Alarm/event printer.

The communication logging is set-up in the bottom and top right section. You can select the protocol(s) and query for which you want to monitor the communications and the type of data scope messages to be logged.

⚠ Enable communication logging only for debugging as it impacts the system performance. Disable it when it isn't needed.

4.4.2. Flow-X communication

The communication with Flow-X is a web-services based protocol. On initialization, eXLerate sends a request to the Flow-X for all live data tags ID numbers and internal names and the Flow-X returns the list with all internal tags.

Message:

`http://[IPAddress]/tags?fields=35&options=42`

Response:

```
<t id="..." n="..." uid="..." />
<t id="..." n="..." uid="..." />
...
```

With:

- t Flow-X tag
- id Flow-X tag internal ID number
- n Flow-X tag internal name
- uid Flow-X tag engineering unit ID
- v Flow-X tag value

The next request is to retrieve all the values for the tags with the IDs where the names match the eXlerate application addresses.

Message:

```
http://[IPAddress]/tags?RawValues=1&Options=43&Fields=513&IDFilter=...%2C...%2C2...
```

Response:

```
<tags cacheid="..." signature="..." starttime="...">
<t id="..." v="..." />
<t id="..." v="..." />
...
```

The following requests include the last retrieved 'CacheID' which result that the Flow-X only returns the tags and values that changed since.

Message:

```
http://[IPAddress]/tags?CacheId="..."&RawValues=1&Options=43&Fields=513&IDFilter=...%2C...%2C2...
```

Response:

```
<tags cacheid="..." signature="..." starttime="...">
<t id="..." v="..." />
<t id="..." v="..." />
...
```

When writing settings and parameter values, the command includes the request for detailed information on the result of the write action. The Flow-X will return detailed information when the write operation fails.

Message:

```
http://[IPAddress]/writetags?errordetails=1&tag...=...
```

Response:

```
<events />
```



Figure 26 Flow-X communication logging

4.4.3. Modbus communication

The Modbus protocol is a master-slave communication protocol. The Modbus Client / Master device originates the communication, it sends a message to request for data or to set data. When a Modbus Server / Slave device receives a message, it sends back a response.

The tables below list the standard data types and the related function codes to read or write data.

Table 3 Modbus data types

Object	Access	Size	Address	Entity number
Coil	Read + write	1 bit	0 to 65535	00001 to 065536
Discrete input	Read only	1 bit	0 to 65535	10001 to 165536
Holding register	Read + write	16-32-64-... bits	0 to 65535	40001 to 465536
Input register	Read only	16-32-64-... bits	0 to 65535	30001 to 365536

Table 4 Modbus function codes

Object	Read function	Single write function	Multiple write function
Coil	1	5	15
Discrete input	2	-	-
Holding register	3	6	16
Input register	4	-	-

A Modbus message frame consists of device address, function code, and data addresses / data values. It is important to make a distinction between entity numbers and data addresses. Entity numbers combine object type and the location within their corresponding table.

For data communications, the entity numbers are translated into 0-based addresses between 0 and 65535. For example, the holding register with address '0' has entity number '40001', entity number '40100' is holding register with address '99'.

The message to read one or more data points is:

```
Message: TI - PI - ML - ID - FC - AD - NO - CRC
```

```
Response: TI - PI - ML - ID - FC - BC - DT - CRC
```

The message to write a single data point is:

```
Message: TI - PI - ML - ID - FC - AD - DT - CRC
```

```
Response: TI - PI - ML - ID - FC - AD - DT - CRC
```

The message to write multiple data points is:

```
Message: TI - PI - ML - ID - FC - AD - NO - BC - DT - CRC
```

```
Response: TI - PI - ML - ID - FC - AD - NO - CRC
```

With:

- TI Transaction ID (Modbus TCP – 2 bytes)
- PI Protocol ID (Modbus TCP – 2 bytes)
- ML Message Length (Modbus TCP – 2 bytes)
- ID Slave device ID
- FC Function code
- AD Starting address (2 bytes)
- NO Number of data points (2 bytes)
- BC Byte count of data
- DT Data of the corresponding points.
- CRC Cyclic Redundancy Check (Modbus Serial)

Should a slave need to report an error, it will reply with the requested function code plus 80 (Hex) (03 becomes 83 in hexadecimal), and will only include one byte of data, known as the exception code:

- 01 Illegal Function code
- 02 Illegal Address Data address
- 03 Illegal Value - not accepted by slave
- 04 Device Failure
- 05 Request is accepted but long duration
- 06 Device Busy. Retry later.
- 07 Negative - cannot perform the functions.
- 08 Parity Error in memory.

Examples:

01 03 00 64 00 14

From device #1, read 16-bits holding registers #100 - 119 (entity numbers 40101 - 40120)

01 03 28 ...

Response with 20x2 data bytes

02 03 00 64 00 14

From device #1, read 32-bits holding registers #100 - 119 (entity numbers 40101 - 40120)

02 03 50 ...

Response with 20x4 data bytes

03 10 00 64 00 02 08 ...

To device # 3, write 32-bits holding registers #100 – 119 (entity numbers 40101 - 40120)

03 10 00 64 00 02

Response data received



Figure 27 Modbus communication logging

4.5. OPC Server

The OPC Server (Data Access 1.0/2.0) is an integrated part of eXlerate, making tag data as found on the xTagDB sheet available to external programs on the computer and/or over a network connection. This communication protocol does not need to be specified explicitly.

The default configuration for eXlerate will not start the OPC Server. If external access to the tag values is desired, an ‘OPCMode’ column needs to be added to the xTagDB sheet, see section ‘Communication fields’ for more details.

When eXlerate OPC Server is running, an extra icon  will be shown in the windows notification area. A user with sufficient privileges (>=2000), can click this icon to open the eXlerate OPC Server monitor window. to view which tag values are currently available in the OPC Server.

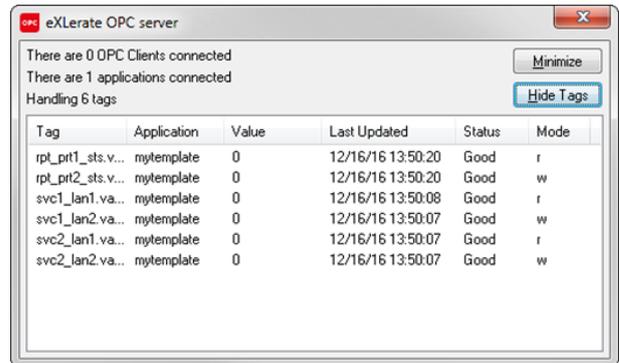


Figure 28 eXlerate OPC Server monitor

5. Tag Database

Tag ID #	Alias	Location	Description	Value	Units	Format	Trend	Query	Address	Data Type	Min	Max	Initial	Scale	Raw Update	OPC Mode	OPC Class	P. Desc	P. Day	P. Month	Weight Factor	Alarm Group	Priority	Behavior	Alarm	Polarity	Minimum	Maximum	Color
1	SVI1_LAN1		Server 1 Network status link 1	1																		Supervisory	1	0					
2	SVI1_LAN2		Server 1 Network status link 2	1																		Supervisory	1	0					
3	SVI1_CPU_USE		Server 1 CPU usage	0.00	%	0.00	0.00%															Supervisory	1	0		0	0	0	
4	SVI1_MEM_USE		Server 1 Memory usage	0	MB	0	0															Supervisory	1	0		0	0	0	
5	SVI1_LEA_USE		Server 1 Handle count	0		0	0															Supervisory	1	0		0	0	0	
6	SVI2_LAN1		Server 2 Network status link 1	1																		Supervisory	1	0					
7	SVI2_LAN2		Server 2 Network status link 2	1																		Supervisory	1	0					
8	SVI2_CPU_USE		Server 2 CPU usage	0.00	%	0.00	0.00%															Supervisory	1	0		0	0	0	
9	SVI2_MEM_USE		Server 2 Memory usage	0	MB	0	0															Supervisory	1	0		0	0	0	
10	SVI2_LEA_USE		Server 2 Handle count	0		0	0															Supervisory	1	0		0	0	0	
11	NET1_STS		Network Printer 1 status	1																		Supervisory	1	0					
12	NET2_STS		Network Printer 2 status	1																		Supervisory	1	0					
13	NET1_LAN1_STS		Communication status link 1	1																		Supervisory	1	0					
14	TIME_CURRENT_TIME		Current date and time	---		---	---	---		1-2												Supervisory	1	0					
15	FORML_STS_COMPANY		Company	---		---	---	---		1-2												Supervisory	1	0					
16	FORML_STS_LOCATION		Location	---		---	---	---		1-2												Supervisory	1	0					
17	FORML_STS_TAG		System tag	---		---	---	---		1-2												Supervisory	1	0					

Figure 29 eXlerate tag database ‘xTagDB’

The tag database sheet **xTagDB** is the beating heart of an eXlerate application. It contains the configuration for ‘tags’ and the live values from the communication with external devices. The word ‘database’ is used as a general classification, i.e. the tag database is arranged as a table, with records (rows) and fields (columns), it is not a relational database.

Each row in the tag database corresponds to one tag. Each column represents a field / property for the tag:

- Naming & reference
- Communication settings
- Live values & units
- Alarm settings
- Trending
- Periodic Averaging & Latching

You may add your own columns / fields, if needed.



Run the *Tag & Object wizard* after making changes to the tag database.

5.1. Tag naming & reference fields

The following fields (columns) define the naming and descriptions of a tag:

- **Group**
(optional) To group tags per query or device.
- **ID**
Unique reference number, used internally. These numbers are automatically generated when running the Tag & Object Wizard.
- **TagName**
Unique name for the tag. It defines the internally used cell and object names. The tag name may be looked at as a ‘key’-field in a relational database.

- **Alias**
(optional) Alternative name to display in alarms & trends. Usually, alias names are longer than the tag name, and it may contain characters that are not allowed for tag names. When an alias name is not defined, the tag name is automatically used.
- **Location**
(optional) Name for the (physical / device) location of the tag.
- **Description**
Full description of the tag. It is used in alarming, and for trend pen selection.

5.2. Communication fields

The following fields (columns) are used to setup the communication and are used in conjunction with the Protocol Table and the Query Table.

- **Query**
ID of the related query/queries used for reading / writing values. It supports redundancy – you can enter multiple queries for a single item separated by commas.
- **Address**
Specific address (Flow-X name, Modbus register, OPC name, HART address) in the query that corresponds with the value for this tag.
- **Data Type**
The data is transmitted in bits. The data type determines how to interpret these bits: as whole numbers, decimal numbers, date and time or text. The data types are predefined constants in eXlerate. The list below contains the most used data types.

Data type	#	Bits	Description
xBit	1	1	Boolean
xByte	2	8	Unsigned integer
xShort	3	16	Signed integer
xWord	4	16	Unsigned integer
xUInt24	5	24	Signed integer
xLong	6	32	Signed integer
xDWord	7	32	Unsigned integer

Data type	#	Bits	Description
xFloat	16	32	Single precision IEEE floating point, standard byte order 43 21
xRevFloat	17	32	Single precision IEEE floating point, reversed byte order 21 43
xDouble	18	64	Double precision IEEE floating point
xShortFloat	19	16	Integer scaled float
xIntelFloat	20	32	Single precision IEEE floating point, byte order 12 34
xRevDouble	22	64	Double precision IEEE floating point byte order 21 43 65 87
xBCD	32	32	BCD value, with 8 times 4-bit, each coded 0..9
xTimeDate	33	64	8-byte date time, format: <YY><MM><DD><hh><mm><ss><uint> YY: Year (0-99) MM: Month (1-12) DD: Day(1-31) hh: Hour (0-23) mm: Minute (0-59) ss: Seconds (0-59) uint: Added value 0-65535
xTimeStamp	34	64	8-byte date time, format: <1><YY><MM><DD><1><hh><mm><ss> YY: Year (0-99) MM: Month (1-12) DD: Day(1-31) hh: Hour (0-23) mm: Minute (0-59) ss: Seconds (0-59)
xAdcFloat	37	12	Value 0-4095 scaled float
x10kFloat	38	16	Value 0-10000 scaled float
xString6	64	48	8-character packed ASCII string (HART)
xString12	65	96	16-character packed ASCII string (HART)
xString24	66	144	32-character packed ASCII string (HART)
xString10	67	80	10 characters string
xString80	68	640	80 characters string
xString	69		A null-terminated string
xString8	70	64	8-byte character string
xString16	71	128	16-byte character string

– **Initial**

(optional) Value of the tag at system startup, until communications are started.

– **Min / Max**

(optional) For scaling data types *xShortFloat*, *xAdcFloat*, *x10KFloat*, or *xWordFloat* and limits used for simulation.

– **ScaleMin/ScaleMax**

(optional) Scale device values to internal values
 $TagValue = ScaleMin + DriverValue * (ScaleMax - ScaleMin)$
 $DriverValue = (TagValue - ScaleMin) / (ScaleMax - ScaleMin)$

– **Update**

(optional) Function to write values to the

device. The eXlerate functions *exUpdateEx(..)*, *exUpdateVarEx(..)*, *exUpdateStrEx(..)* update the *xIConnect* component, that will write the new value to the device.

– **OPCmode**

(optional) Given you have an OPC server license, you can specify tags to be available in the eXlerate OPC Server and accessible by other OPC Clients. To enable tags in the OPC Server, simply add an “R” (read only), “W” (write and read) or “H” (hidden) in the OPCMode column for the specific tag.

Note that accessibility may depend on DCOM security settings on both computers.

– **OPCgroup**

(optional) This column will supply an OPC group name for the tag. If it is empty, the ‘Location’ is used as OPC group name.

5.3. Values fields

The value fields contain the live values when communications are running. Normally, the ‘Value’ cells are unprotected, unless a formula is entered in a cell rather than a straight value.

The tag database contains following value fields:

– **Value**

Cells for the(live) value from the devices.

Running the Tag & Object Wizard creates the cell names like: ‘*xMyTag.Value*’.

– **Value2, ..., Value5**

(optional) For redundancy, the Value2...Value5 may be used: a single eXlerate tag uses data from various parallel running devices. The “Value” is used for alarming, trending, etc.; The others are used for selection and comparison. Running the Tag & Object Wizard creates the cell names like: ‘*xMyTag.Value2*’.

– **WriteValue**

(optional) Value to write to the device, used in combination with the **Update** column.

– **Units**

The engineering units as used in the application for this tag.

Running the Tag & Object Wizard creates the cell names like: “*xMyTag.Units*”.

5.4. Alarm fields

Alarms inform users about problematic conditions such as faulty equipment or out-of-range values.

The tag database contains the following columns for the alarm manager:

- **AlarmDesc**
Text to show for the alarm. When omitted, the tag “Description” field is used.
- **AlarmGroup**
Group to which the alarm belongs to. Alarms may be grouped (hierarchical), like a directory tree structure. The “Location” field is used when omitted.
- **Priority**
Alarm priority used for filtering and sorting alarms in the user interface.
- **SAlarm**
Defines a status alarm associated to the tag. Usually associated with digital signals – enter “0” or “1” to indicate the alarm status. Running the Tag & Object Wizard creates the cell names like: ‘*xMyTag.SAlarm*’.
- **LLAlarm**
Defines the 2nd low level for the tag. The alarm is active when the value is below this limit. Running the Tag & Object Wizard creates the cell names like: ‘*xMyTag.LLAlarm*’.
- **LAlarm**
Defines the 1st low level for the tag. The alarm is active when the value is below this limit. Running the Tag & Object Wizard creates the cell names like: ‘*xMyTag.LAlarm*’.
- **HAlarm**
Defines the 1st high level for the tag. The alarm is active when the value is above this limit. Running the Tag & Object Wizard creates the cell names like: ‘*xMyTag.HAlarm*’.
- **HHAlarm**
Defines the 2nd high level for the tag. The alarm is active when the value is above this limit. Running the Tag & Object Wizard creates the cell names like: ‘*xMyTag.HHAlarm*’.
- **Deadband**
The minimal change before a limit alarm returns to “normal”. This is to prevent jittering alarms when the value is close to the limit. Running the Tag & Object Wizard creates the cell names like: ‘*xMyTag.Deadband*’.
- **Delay**
Delays activation/de-activation of the alarms: the new condition should exist for this period (seconds), before the alarm manager generates an event.

Running the Tag & Object Wizard creates the cell names like: ‘*xMyTag.Delay*’.

5.5. Trend fields

With eXlerate comes the ability to trend live values over time. You configure the following fields for tags that you want to be trended:

- **TrendNorm**
Trend data is stored on value changes. This field defines the (minimum) change for the value required, before it is stored.
- **Format**
The format to show the data on the trend labels (like “0.00”).

5.6. Periodic fields

An eXlerate application can perform periodic average calculations and periodical latch live values. These calculations are created automatic from the tag database, using the following fields:

- **WeighFactor**
Tag to use as weight factor for averaging. The field should contain a reference name (*xMyTag.Value*), not a reference itself (*=xMyTag.Value*). The reference should be an incrementing value, like a flow total. Use *xNow.Time* for time-weighted averaging.
- **P_[name]**
Period related fields to define if values should be averaged/latched for this period. To create average values for a named period, enter a “W” in this field. To periodically latched values, enter “L”.

The configuration and calculations are described in more details in section *‘Interval periods and events’*.

5.7. Tag count

Each row in the tag database corresponds to one tag. The number of tags that you may have in your application depends on your license. If you have more tags (rows) in the *xTagDB* than your license, only the tags corresponding to the license will be updated.

6. Calculations

Most probably you want to create your own calculations, based on a combination of (live) values, and use these on displays and reports in your application. As example, you want to calculate the total station flow rate based on the flow rates of the individual streams or you want to determine a valve position out of two digital (open/closed) tag values.

The calculation sheets provide the location to put these custom calculations and additional (modifiable) parameter values. An application can have multiple calculation sheets, each sheet containing multiple calculations.

6.1. Calculation sheets

All calculation sheet names start with “**xCalc**” and have a “header” row on row #3 defining the fields (columns) for the calculations on that sheet.

Calculations				
Group	CalcTag	Description	Value	Store
Station			Station	
	Calc.Stn_GVR_CUR	Gross volume flow rate	108.008116	
	Calc.Stn_BVR_CUR	Base volume flow rate	4253.79331	
	Calc.Stn_MASSR_CUR	Mass flow rate	2891.892618	
	Calc.Stn_ENGYR_CUR	Energy flow rate	160.7151828	
	Calc.Stn_TT_CUR	Temperature	40.0022536	
	Calc.Stn_PT_CUR	Pressure	4000.22536	

Figure 30 Calculation sheet

- **Group**
(optional) name for grouping calculations to structure your application.
- **CalcTag**
Unique name for the calculation tag. By convention, all calculation names start with “**Calc.**”. The Calculation wizard uses this name for generating cell names.
- **Description**
Descriptive text explaining the calculation.
- **_Value**
The ‘_Value’ fields, starting with an underscore, contain the formulas for the calculated value, or custom parameter values.
Running the Calculation Wizard creates the cell names like: ‘Calc.MyTag’.
- **_[Text]**
Other fields starting with an underscore, can contain additional calculations / parameters
Running the Calculation Wizard creates the cell names like: ‘Calc.MyTag’_[Text].

- **Store**
When using parameters on calculation sheets. See section ‘*Store values*’.
- ...
Other fields (columns), not starting with an underscore, can contain additional descriptions, references to other input values, or other functions or intermediate calculations.

In the calculation you can use

- references (named) cells, like
=D5+D6+D7
=xSTR1_GVR_CUR.Value+xSTR2_GVR_CUR.Value
- excel functions, like
=AVERAGE(D5:D7)
=VLOOKUP(D6,E5:F8,2)
- eXlerate functions, like:
=exInterpolate(E5:E8,F5:F8,D8)
=fxAGA8_M(,xPT.Value,xTT.Value,xGAS_COMP)
- Your own User Defined Function from VBA, like
=MyVbaFunction(...)



Don't use volatile functions in your application, but their alternatives – see section ‘*Formulas*’.

6.2. Store values

The eXlerate function ‘*exStoreValue()*’ saves a value when it changes. The value will be retentive and loaded back into the application start. The value is synchronized on redundant systems.

When using *exStoreValue()* the value is considered a parameter. Any change is stored into the event logger.

These functions are normally put in the ‘*Store*’ column on calculation sheets.

7. Displays

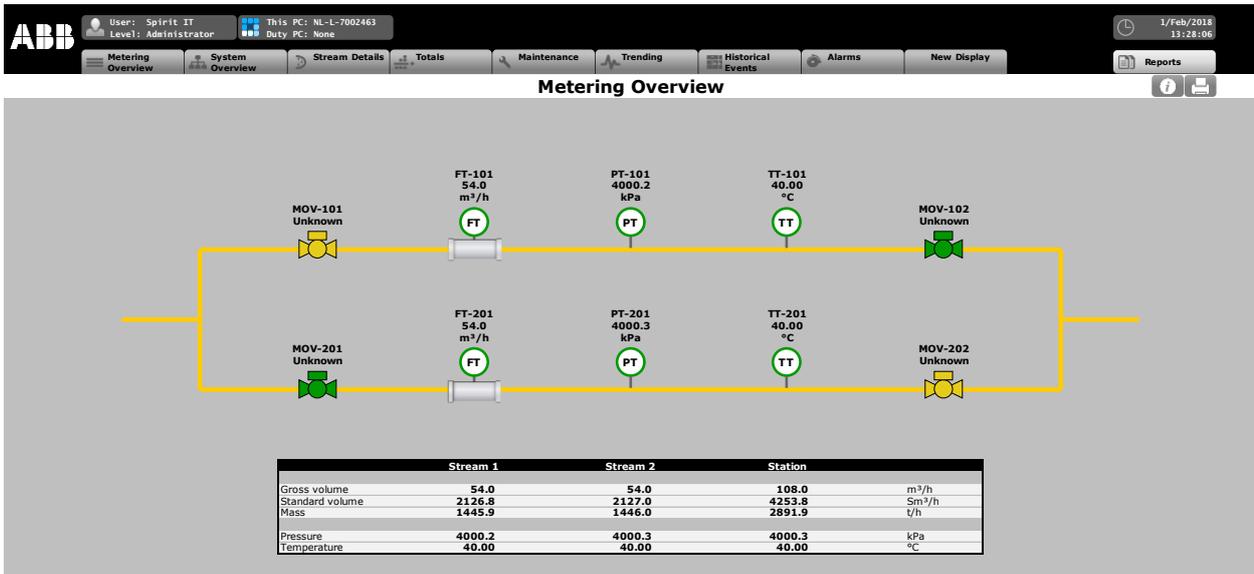


Figure 31 Application display example

The User Interface, or Human Machine Interface (HMI), of your application consists of several displays. Each display is configured as a separate Excel sheet in an application.

Display sheets can contain:

- Cells with (fixed) text
- Cells referring to live/calculated values
- Numerical formatting for values
- Pictures
- Animations based on live values
- Charts
- Alarm controls
- Trend controls

7.1. Configuration tables

The worksheet **xTables** contains the configuration tables. The following tables are used for configuring displays:

- User table
- Worksheet table
- Style table
- Color table
- Button table

Animations of shapes and ranges are defined in the **xAnimations** sheet.

7.1.1. User table

The security levels in your application is based on numerical values, linked to the user levels as in the eXlerate Control Center (see section 'eXlerate user accounts'). When a user logs in, the Control Center checks the username and password and assigns the numerical security level.

The user table defines the numerical levels as named user groups so you can refer to these group names instead of numerical values when defining security levels in your application. This table contains three columns:

- **Level**
Numerical value for the user group
- **GroupName**
Internal name of user group level
- **Comments**
Description of the user group

UserTable		
Level	GroupName	Comments
0	Accesslevel.Guest	Guest
500	Accesslevel.Operator	Operator
750	Accesslevel.Technician	Technician
1000	Accesslevel.Engineer	Engineer
1500	Accesslevel.Supervisor	Supervisor
2000	Accesslevel.Administrator	Administrator

Figure 32 User table



Run the 'Tag & Object Wizard' after making configuration changes.

7.1.2. Worksheet table

As seen in section ‘eXlerate worksheets’, the worksheets in an eXlerate application have different functionality. You define which sheets are display sheets and are available and visible in *Runtime* mode in the ‘Worksheet table’. Only sheets defined in this table are visible in Runtime mode.

WorksheetTable Display size 1920 x 1080			
Worksheet	Visible level	Edit level	UIRange
sStnOvw	0	0	A1:Q59
sSysOvw	0	0	A1:Q59
sTotals	0	0	A1:Q79
sMaint	0	0	A1:R59
sLegend	0	0	A1:Q59
sTrends	0	0	A1:Q59
sAlarms	0	0	A1:Q59
sEvents	0	0	A1:R59

Figure 33 Worksheet table

The ‘Worksheet table’ defines the following properties for the display sheets:

- **Worksheet**
Name of the sheet that is a display.
- **Visible level**
Minimum user level required to show the display.
- **Edit level**
Minimum user level required to edit values on the display in runtime. Available for backward compatibility – always set to “0” (zero).
- **UI range**
User interface range that is visible/editable.



– Run the ‘Tag & Object Wizard’ after making the configuration changes.

7.1.3. Style table

The ‘Style table’ is used to define the format/style of displaying (live) values and units on displays and reports, and scaling for communications.

StyleTable				
Name	Format	Units	Scale	Offset
Date	d/mmm/yyyy			
Time	HH:mm:ss			
DateTime	d mmm 'yy HH:mm			
FullDateTime	/mmm/yyyy HH:mm:ss			
Number		0		
Percentage		0.00 %		
DifferentialPressure		0.000 kPa		
Frequency		0.00 Hz		
Temperature		0.00 °C		
Pressure		0. kPa	100	0

Figure 34 Style table

The ‘Style table’ contains the follow fields for each entry:

- **Name**
Unique name for the format style.

- **Format**
Formatting for values – reference: “xf_[Name]”
- **Units**
Engineering units – reference: “xu_[Name]”
- **Scale**
Scaling for values (for use in xTagDB) – reference: “xs_[Name]”
- **Offset**
Offset for values (for use in xTagDB) – reference: “xo_[Name]”

After modifications, the cell styles are updated as defined in this table. Styles that are not listed in this table are removed from the list.

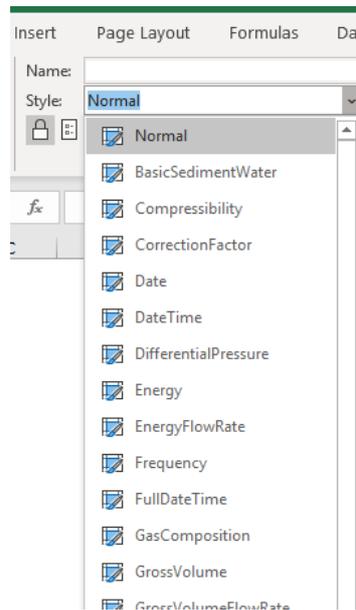


Figure 35 Excel style selection list

7.1.4. Color table

In eXlerate, 64 colors are available to use for animating shapes in runtime based on (live) values. The ‘Color table’ defines the colors in **R**ed, **G**reen, **B**lue values. You can use these colors for animations by using the color ID (number 0 – 63).

ID	R	G	B	Comment
0	0	0	0	0 Fixed color: 'xBlack'
1	255	255	255	255 Fixed color: 'xWhite'
2	255	0	0	0 Fixed color: 'xRed'
3	0	255	0	0 Fixed color: 'xGreen'
4	0	0	255	255 Fixed color: 'xBlue'
5	255	255	0	0 Fixed color: 'xYellow'
6	255	0	255	255 Fixed color: 'xMagenta'
7	0	255	255	255 Fixed color: 'xCyan'
8	0	0	0	0 Design color black
9	255	0	0	0 Design color red
10	0	153	0	0 Design color green
11	0	0	204	204 Design color blue
12	255	255	255	255 Design color white
13	230	76	0	0 Design color orange
14	255	255	0	0 Design color yellow
15	204	204	204	204 Design color yellow
16	51	102	153	153 ...
17	230	230	230	230 Background
18	0	0	51	51 Text color 1
19	255	255	255	255 Text color 2
20	51	102	153	153 Table color 1
21	75	135	195	195 Table color 2
22	151	186	226	226 Table color 3
23	153	153	153	153 Flow unknown

Figure 36 Color table

The first 8 colors are fixed. You can refer to these colors by using the color ID (number 0 – 7) or the color names:

- 0. xBlack
- 1. xWhite
- 2. xRed
- 3. xGreen
- 4. xBlue
- 5. xYellow
- 6. xMagenta
- 7. xCyan

The other 56 colors are user definable. You can change these colors by setting the **Red**, **Green**, **Blue** values between 0 (absent) and 255 (fully present).



Run the 'Color wizard' when you have made changes to the Color table.

7.1.5. Button table

You can add buttons on your displays to perform actions in runtime, like navigate through the displays, send commands to devices (open valve, start new batch). The 'Button table' contains the configurations of the button objects.

Button Table							
Worksheet	Button	Text	Key	Procedure	Macro/Command	Access	Enabled
			^j	exShowLoginDialog	Call_exShowLoginDialog	0	TRUE
			^(BS)	exLoadPrevPage	Call_exLoadPrevPage	0	TRUE
			^(Home)	Load_sMainOvw	Call_Load_sMainOvw	0	TRUE
	btnLogin			exShowLoginDialog	Call_exShowLoginDialog	0	TRUE
	btnInfo		^i	Load_sLegend	Call_Load_sLegend	0	TRUE
	btnPrintPage		^p	exPrintCurrent	Call_exPrintCurrent	500	TRUE
	btnReports	Reports	^r	xfReports.Show	Call_xfReports_Show	500	TRUE
	btnDateTime		^t	SetSystemDateTime	Call_SetSystemDateTime	999	TRUE
	btnComputers			xNet_exDutySwitch	Call_xNet_exDutySwitch	999	TRUE
	Button1	Metering Overview	{F1}	Load_sMainOvw	Call_Load_sMainOvw	0	TRUE
	Button2	Metering Overview	{F1}			0	TRUE
	Button3	System Overview	{F2}	Load_sSysOvw	Call_Load_sSysOvw	0	TRUE
	Button4	Stream Details	{F3}			0	TRUE
	Button5	Totals	{F4}			0	TRUE
	Button6	Maintenance	{F5}			0	TRUE
	Button7	Trends	{F6}	Load_sTrends	Call_Load_sTrends	0	TRUE
	Button8	Historical Events	{F7}	Load_sEvents	Call_Load_sEvents	0	TRUE
	Button9	Alarms	{F8}			0	TRUE
	Button10		{F10}			0	TRUE
	sLegend	btnInfo	^i	exAboutBox	Call_exAboutBox	0	TRUE
	sEvents	btnPrintPage	^p	sEvents.PrintReport	Call_sEvents_PrintReport	500	TRUE
	sTrends	btnSave	^s	xfTrendExport.Show	Call_xfTrendExport_Show	500	TRUE
	sSysOvw	btnExplorer	^e	WinExplorer	Call_WinExplorer	999	TRUE
	sSysOvw	btnCtrlCenter	^c	ShowControlCenter	Call_ShowControlCenter	999	TRUE
	sSysOvw	btnDiagnosics	^d	xfDiagnosics.Show	Call_xfDiagnosics_Show	999	TRUE
	sSysOvw	btnQuit	^q	QuitApplication	Call_QuitApplication	999	TRUE
	sSysOvw	btnTagDB		Load_xTagDB	Call_Load_xTagDB	1500	TRUE
	sSysOvw	btnComms		Load_xComms	Call_Load_xComms	1500	TRUE
	sMainOvw						TRUE
	sTrends						TRUE

Figure 37 Button table

The 'Button table' contains a row (entry) for each button in your application. For each entry, you can set-up the following configuration fields:

- **Worksheet**
Name of the sheet where the button is located. Button definitions of the same worksheet are grouped together. Leave empty if it is applicable for all displays.
- **Button**
Name of the shape that acts as a button.
- **Text**
Text to be placed on the button.

Use a tilde ('~') character in the text for a new line. The first line of the text will be placed bold, the rest of the text in plain font.

- **Key**
Key stroke (combination) associated with the button. Special keys are defined between curly brackets '{' and '}'. The following special keys are available:

- {F1} ... {F15}
- {LEFT}
- {RIGHT}
- {UP}
- {DOWN}
- {PGUP}
- {PGDN}
- {HOME}
- {END}
- {INSERT}
- {BACKSPACE} or {BS}
- {DELETE} or {DEL}
- {CLEAR}
- {CAPSLOCK}
- {NUMLOCK}
- {BREAK}

Key codes may be also used in combination with the <Alt>, <Shift> and/or <Ctrl> keys. In that case, the character key is preceded with the following key state character codes:

- <Shift>: '+' (plus sign)
- <Ctrl>: '^' (caret sign)
- <Alt>: '%' (percent sign)

For example: '+{PgUp}'

- **Procedure**
VBA procedure to execute when the button or key combination is pressed. You need to enter either a predefined eXlerate procedure or your own defined VBA procedure or user form that is available in your application.

The following procedures are predefined:

- Load_{SheetName}
Show the specified display (sheet).
- exLoadPrevPage
Show the previous loaded display (sheet).
- exPrintCurrent
Print the current display (sheet).
- exShowLoginDialog
Show the Login/Logout dialog.
- exAboutBox
Show the 'About' box of eXlerate
- AcceptEditGroup_{ }
Accepts the new values for a group of editable cells (see section 'Editing values').

- **Macro/Command**
Automatically generated by the Button Wizard:
Actual macro being executed including the security check for the procedure to execute.
- **Access Level**
user access level required to perform the action assigned (press the button).
- **Enabled**
Always set the value 'TRUE' to enable the button.



Run the 'Button wizard' when you have made changes to the Button table.

7.2. Display sheets

Each runtime display is a separate sheet in the application. You must define which sheets are display sheets in the 'Worksheet table'.

When you start with an application, first determine the display resolution of the runtime computer system and set this resolution for the application. It is time-consuming to convert your displays to a different resolution at a later stage.

To avoid repetitive work, it is recommended you define a 'Template' sheet, that contains the default look of all your displays, including default navigation buttons, logo's and other objects. The objects on this 'Template' sheet only need to be defined once and can be updated automatically on all your displays, see section 'Button wizard'.

7.2.1. New display sheets

To create a display sheet, just make a copy of the 'Template' sheet. Right click the worksheet tab, enable "Create a copy" checkbox, select "Move or Copy..." and rename the copied sheet (preferable starting with a lowercase "s").

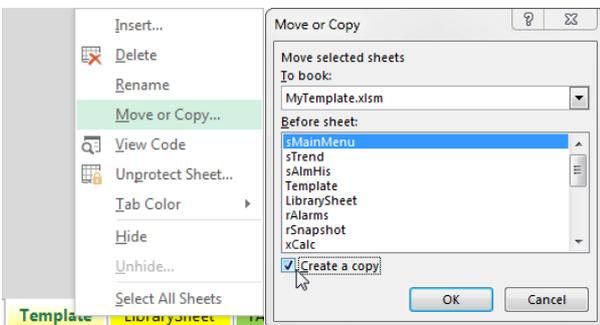


Figure 38 Copying the Template Sheet

The final thing to do is to register the new display in the 'Worksheet table'.

7.2.2. Text, live values and units

To show (fixed) text on a display, just type the text into a cell. For values available on (other) sheets in the application, e.g. live / calculated values in the tag database / calculation sheets, type the reference to the cell name, like "=xMyTag.Value".

To set the numerical format, select the cell and select the style format from the ribbon Style box. Alternatively, you can use the excel cell format window (press <Ctrl-1>).

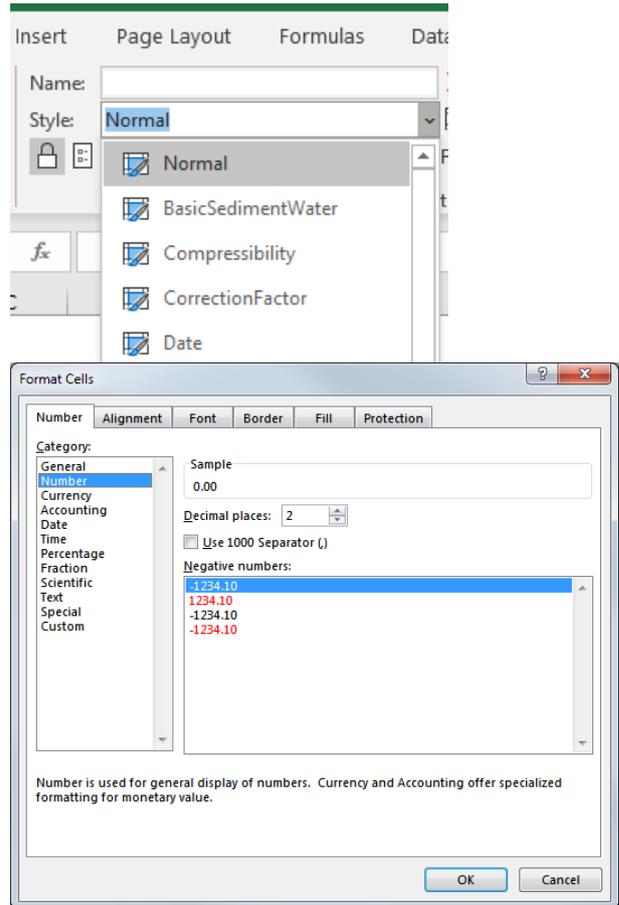


Figure 39 Set cell numerical style format

For engineering units, it is recommended that you refer to the 'Style table' units ("=xu_MyUnits") or the units in the tag database ("=xMyTag.Units"). This enables to have one location to change the units, so you don't need to check all displays when you need to change the units. Alternatively, you still can type in fixed text.

The background fill and the text color for cells can be changed based on live values and calculations. If you want ranges to be animated, give the range a name, using the *Name input*.

7.2.3. Charts

Standard excel charts are available to put on your display sheets. These charts will change when the source data changes. When you refer to live / calculated values, the chart will be dynamic in runtime mode. It is recommended for charts that you refer to data ranges on the same sheet as the chart is placed.

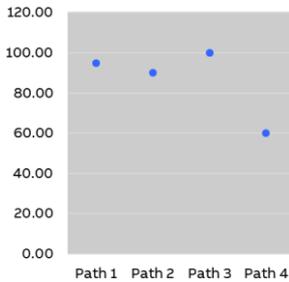


Figure 40 Standard chart for displays

7.2.4. Pictures

You can put pictures, like a company logo, on displays. Simply select the *Insert - Picture* from the ribbon and browse for the file.

7.2.5. Shapes

To visualize the field layout and status, you can use (basic) shapes, like lines, rectangles, ovals, arrows, cylinders, and much more. With shape objects you can draw almost anything, piping, transmitters, valves, tanks – your system P&ID.

You can add shapes to your display by selecting them from the ribbon *Insert - Shapes* or copy and paste existing ones.

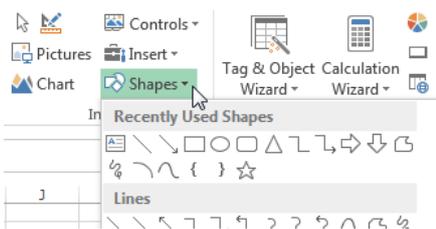


Figure 41 Using the Shapes Button

Shapes can be grouped together to form new shapes using the *Format* menu.

You can modify the look & feel of a shape using the *Format Picture* window (<Ctrl-F1>) and the *Shape selection* window (<Ctrl-F10>).

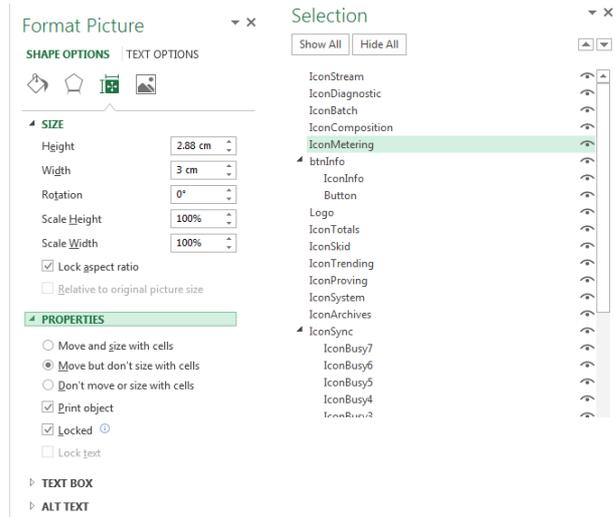


Figure 42 Select and format shapes

When you have added a shape, you should give the shape a name using the *Name input* on the eXlerate ribbon. Note that the *Name Definition Tool* is not available for shapes.

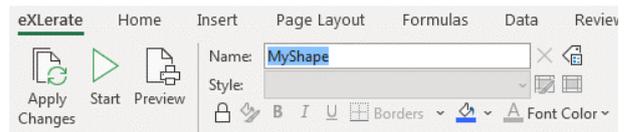


Figure 43 Set shape object name

Once a shape has a name, you can use it in animations - see below. If shapes are grouped together, the sub-shapes can be accessed using a ‘.’-character as separator: two (child) shapes named ‘*Shape1*’ and ‘*Shape2*’ grouped together into a (parent) shape named ‘*MyGroup*’ can be accessed by names ‘*MyGroup.Shape1*’ and ‘*MyGroup.Shape2*’.

⚠ Limit the grouping of shapes to one (1) level to allow animations for the sub-shapes.

7.3. Animations

In eXlerate, you can animate shapes and cells (ranges), e.g. the properties of a shape / cell are automatically changed based on live values and calculations.

For shapes, the following properties can be animated:

- Fill color
- Line color
- Blink between two colors
- Show or hide the object
- Position on sheet (in points from left-top)

- Width (in points)
- Height (in points)
- Rotation angle

For ranges (cells)shapes, the following properties can be animated:

- Fill color
- Text color
- Blink between two colors

7.3.1. Animation object names

Names used in an application have a global scope. A global scope means that all occurrences of all shapes in all display pages with this global name are animated identically: if you have two shapes with the same name on different sheets, you only define one entry in the animation table. These shapes will be animated identically using the same configuration in the animation table.

7.3.2. Configuration

The animations are defined on the **xAnimations** worksheet. Each row contains the configuration for one entry (shape object or cell range). When you are using a recent template application, shapes and named ranges on display sheets may be automatically added when activating the sheet. You always can add entries manually to the sheet.

Figure 44 Animations Table

The columns of a row are the configuration fields and are divided in 4 sections. You need to set-up the animations for each entry using the following fields:

Identification of animation object

- **Class**
Use for group sections of objects for application maintainability purposes.
- **Shape/Range**
Name of the shape or range to animate.
- **Description**
Text describing the object / animation

User calculations

- **Value...**
Your own references and (pre-) calculations of values to use for live animation.

Animation values

Actual (calculated) values to use for animations, based on live / (pre-)calculated values.

- **Fill color**
The shape / range fill color - a number referring to the Color table ID.
- **Line/Text color**
The shape line color / range text color - a number referring to the Color table ID.
- **Blink color**
The color to use as blink color - a number referring to the Color table ID.
- **Blink**
Enable (TRUE) or disable (FALSE) blinking.
- **Visible**
Show (TRUE) or hide (FALSE) the object.
- **Left / Top.**
Object position (in points) form top-left corner.
- **Width / Height**
Object size (in points).
- **Rotate**
Rotation (in degrees) of the object.

The colors used for animations are the 64 colors of the eXLerate Color table.

eXLerate animation functions

The last section contains the eXLerate functions that trigger the animations in runtime according the animation values above. These functions are described in more detail in the 'Function Reference' help file. For shapes, the functions start with *'exShape...'* For ranges, the functions start with *'exRange...'*

When adding entries manually, copy the eXLerate animation functions (in the grey columns) from existing rows as these functions should exist for each object.

7.4. Buttons & navigation

You can add buttons to your application displays to navigate through the different displays and to perform your own control actions, like starting a proving sequence.

To add a button to a display, simply add a shape to the display and give it a (logical). Then create an entry in the Button table, configure the fields for the buttons and run the *'Button wizard'*.

When you have created a template sheet, you can add the buttons that are the same for all displays (e.g. display navigation buttons) to this Template sheet. You then only need to define these buttons once.

8. Alarm management

Alarms inform users about problematic conditions such as faulty equipment or out-of-range measurements. Alarms need to be acknowledged – even when a problematic condition disappears. Alarms are shown prominently in the user interface. An alarm remains visible until the user has indicated that he is aware that it has happened.

In eXLerate you configure alarms in design, and you define the user interface(s) to monitor alarms and control alarms in runtime. For the latter one, a simple to configure control is available.

8.1. Defining alarms

Alarms are defined in the tag database. See section ‘Alarm fields’ for the columns defining the alarms.

Run the **Tag& Object wizard** after changes to the tag database. It automatically generates the names for tags with alarm properties defined:

- x{TagName}.SAlarm
Status alarm limit
- x{TagName}.LLAlarm
Low-Low alarm limit
- x{TagName}.LAlarm
Low alarm limit
- x{TagName}.HAlarm
High alarm limit
- x{TagName}.HHAlarm
High-High alarm limit
- x{TagName}.Deadband
Deadband value for limit alarms
- x{TagName}.Delay
Delay for alarms
- x{TagName}.AlmCount
Number of active alarms
- x{TagName}.AlmUnack
Number of unacknowledged alarms

For each status / limit alarm above, the wizard generates two additional names:

- x{TagName}.[S/LL/L/H/HH]Alarm.Raised
A boolean that indicates if an alarm is active
- x{TagName}.[S/LL/L/H/HH]Alarm.Status
Alarm state value as listed in the table below.

Table 5 Alarm state values

Value	Status description
-6	Alarm status undefined
-5	Alarm is active and unacknowledged
-4	Alarm is active and unacknowledged
-3	Alarm is active and blocked (too many alarm changes without acknowledgement)
-2	Alarm is inactive and blocked (too many alarm changes without acknowledgement)
-1	Alarm is inactive and unacknowledged
0	Alarm is inactive and acknowledged
1	Alarm is active and suppressed (by operator)
2	Alarm is inactive and suppressed (by operator)
3	Alarm is disabled (by the system)

8.2. Alarm groups

Alarms can be grouped hierarchically way with ‘parent’ and ‘child’ groups. This will show these alarms in a tree structure in the user interface. Users can view and acknowledge alarms in a group at once. You define the alarm groups in the ‘Alarm Groups Table’ on the ‘xTables’ sheet.

AlarmgroupsTable		Alarm Groups
Parent	Child	
System	Supervisory	<input type="radio"/> All
System	Stream 1	<input type="radio"/> Flow-X
Stream 1	Flow stream 1	<input type="radio"/> Stream 1
Stream 1	Temperature stream 1	<input type="radio"/> Flow stream 1
Stream 1	Pressure stream 1	<input type="radio"/> Pressure stream 1
System	Stream 2	<input type="radio"/> Temperature stream 1
Stream 2	Flow stream 2	<input type="radio"/> Stream 2
Stream 2	Temperature stream 2	<input type="radio"/> Flow stream 2
Stream 2	Pressure stream 2	<input type="radio"/> Pressure stream 2
		<input type="radio"/> Temperature stream 2
		<input type="radio"/> Supervisory

Figure 45 Alarm groups table

This table consists of two columns: ‘parent’ and ‘child’. The root node of the alarm group tree is the “System” group. The first level groups are a ‘child’ of this “System” group: add these to the ‘Alarm Group table’ with “System” in the ‘Parent’ column and the group name in the ‘child’ column. You can add subgroups by adding the name of the main group in the ‘Parent’ column and the subgroup in the ‘child’ column. Note that each group name can only have one ‘parent’.

To add an alarm to an alarm group, take the group name that you defined in the ‘Child’ column of the ‘Alarm Group Table’, and insert it into the ‘AlarmGroup’ column in the ‘xTagDB’ sheet.

After running the Tag & Object wizard alarms are sorted on a hierarchically way with parent and child groups.

 Run the **Tag & Object Wizard** after making changes to the alarm groups.

8.3. Active alarms

The intend of alarming is to inform users in runtime about problematic conditions. The ‘Alarm summary control’ is the eXlerate control that you can use to make the current alarm statuses visible and allow user interaction in runtime. This control is easy to implement. You can add an alarm summary to a display sheet, just select it from the Controls menu.

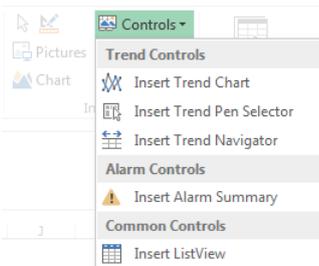


Figure 46 Insert Alarm Summary Control

When you inserted a new control, you can position and resize it on the sheet. Make sure the  option from the eXlerate menu “Insert” section is enabled (default after inserting a control).



Figure 47 The Alarm Summary Control

The Alarm Summary Control is easy customizable. In Design mode, you can change look and feel for runtime systems, like setting:

- Toolbar buttons availability
- Alarm columns visibility, position and size:

ID	Internal alarm ID
State	Icon indicating the alarm status: <ul style="list-style-type: none">  Alarm is active  Alarm is acknowledged  Alarm is blocked  Alarm is suppressed.  Alarm is disabled.
Timestamp	Time of last state change.
Name	Name/alias of the tag.
Location	Alarm group name
Priority	Alarm priority.
Type	Alarm type: <ul style="list-style-type: none"> Status Alarm Low Alarm LoLo Alarm High Alarm HiHi Alarm
Description	Alarm / tag description.
Limit	Current limit value.
Deadband	Deadband value if configured.
LastValue	Last value of tag.
Units	Engineering units.
Delay	Alarm delay if configured.
BlockCount	Active, unacknowledged counter. See ‘Advanced alarm’.
Format	Numerical format of the tag

- Security to acknowledge, edit or suppress alarms
- Colors of the alarm states,
- Add filters for showing alarms

To change these properties, disable the  option from the eXlerate menu “Insert” section and click the  option in the right-top corner of the control.

8.4. Historical events

Alarms & Events are stored in internal database when the application (communication) is running. Many applications feature an alarm history. The ‘MyTemplate’ application contains a pre-defined historical alarms & events display with a ‘ListView’ control. Programmatically (VBA) the historical alarms & events data retrieved from the database table and put into this control in runtime. A user

can select date/time, scroll, sort data with the pre-configured buttons on this sheet.

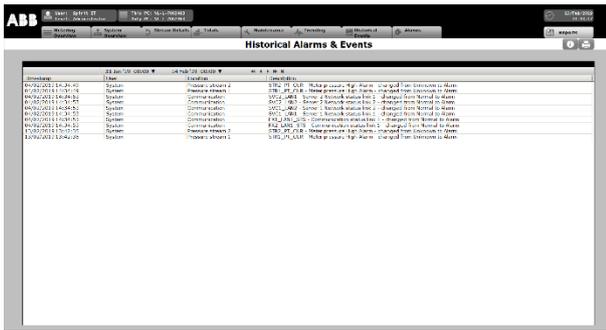


Figure 48 Historical alarms & events

Setting-up such a display requires programming and database skills and is not covered within this document.

8.5. Advanced alarm usage

There are several additional worksheet and VBA functions that can be used for advanced configurations in your application. These functions are described in more detail in the 'Function Reference' help file.

- exSetAlarmDeadband
Programmatically (from VBA) change the deadband value.
- exSetAlarmLimit / exSetAlarmLimit2
Programmatically (from VBA) change an alarm limit without the standard alarm editor.
- exSetAlarmDelay
Programmatically (from VBA) change the alarm delay without the standard alarm editor.
- exAlarmSetModeByID /
exAlarmSetModeByGroup
Programmatically (from VBA) disable or suppress an alarm (group)
- exAlarmShowStatus
Send the all alarms with a certain state (active, disabled, or suppressed) to the system event logger.
- exAlarmCount
Count the number of alarms within a specific group that have a specific state.
- exAlarmAckGroup
Acknowledge all alarms in an alarm group.
- exAlarmDisable
Suppress/unsuppress or disable/enable a single alarm or alarm group
- exAlarmSetOptions.
 - Automatic acknowledgement
When you enable automatic acknowledge alarms, the user does not need to acknowledge alarms.
 - Alarm blocking
Enabling alarm blocking will disable new alarm messages when too many alarm transitions without acknowledgements have taken place, e.g. the user ignores alarm messages.
You can define the amount of times that an alarm changes state before it obtains a 'blocked' status.
Note that blocking has no effect when Automatic acknowledge is enabled.
 - Only show active alarms
 - Only show active or unacknowledged alarms

9. Trending

eXlerate comes with the ability to trend and show values over time in charts. You define the tags that need to be trended in design and create the user interface that allows to select and view these trended values in runtime. eXlerate has Trend Controls that are simple to configure and used for runtime operations to shows trend charts.

9.1. Defining trend tags

You have the tags that need to be trended in Tag Database, using the following two columns:

- TrendNorm
The minimum change to store values in the historical trend database. E.g. 0.1 means if the difference between the current- and previous value is higher than 0.1, the value is stored into the database. When a value of 0 is used, every change is stored into the database.
- Format
The format to show the numerical data into the labels. E.g. “0.0” or “0.000”.

Run the **Tag& Object wizard** after changes to the tag database to update the tags to be trended and the trend norm to use.

9.2. Data storage

Trend data is stored in binary files on disk. The data storage is defined by the tag fields set-up in the tag database and is independent of the trend charts pen selections in the application. So even when no pen is selected, the data is stored according to the trend norm (see above).

For each tag that is to be trended, a separate folder is created in the “TrendData” folder, each containing 3 or more sub-folders. The sub-folders “Raw”, “10Days” and “100Days” are used for storing the actual trend-values, while other sub-folders are used for storing additional data such as limits. These folders are automatically created when necessary. The files with trend data have the extension “xtd” (eXlerate Trend Data File) and the name of a file identifies the start-date of the file based on UTC (=GMT 0):

9.3. Display trends

The user interface for trending in eXlerate is easy to build using three controls:

- Trend Chart
Visualizes the trend-data in a chart, including controls for zooming of the trend-data and moving forward and backward in time.
- Trend Pen Selector
Selection of trend tags for trend chart. This control is linked to a Trend Chart control.
- Trend Navigator
Visualizes a larger portion of the trend-data to quickly see anomalies in the data. This control is linked to a Trend Chart control.

You can add these Trend Controls to your application from the eXlerate “Controls” menu.

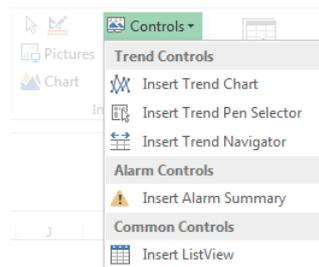


Figure 49 Insert Trend Controls

When you inserted a new control, you can position and resize it on the sheet. Make sure the  option from the eXlerate menu “Insert” section is enabled (default after inserting a control).

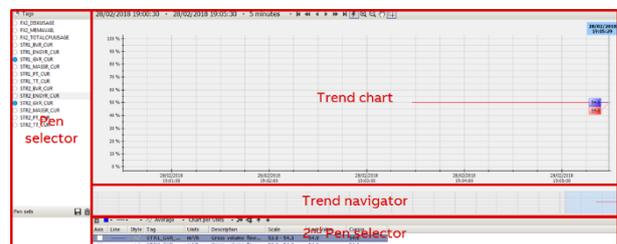


Figure 50 Trend display

You can change look and feel of the controls for runtime systems. To change the properties of a control, disable the  option from the eXlerate menu “Insert” section and click the  option in the right-top corner of the control.

9.3.1. Trend Chart

The main trend chart is the graphical visualization of the (selected) tag values over the selected period. In design mode, you can set properties for the main trend chart:

- Toolbar buttons available in runtime
 - Date/time Start/end date and time to show for the chart. When clicked, a popup is shown for selecting the date/time.
 - ... hour Period of the trend chart.
 - ⏪ Move to the start – first time a value was stored for any of the pens.
 - ⏩ Move fast backward – default step-size is 100%, causing to move back a whole period.
 - ◀ Move backward – default step-size is 25%, causing to move back a quarter period.
 - ▶ Move forward – default step-size is 25%, causing to move forward a quarter period.
 - ⏭ Move fast forward – default step-size is 100%, causing to move forward a whole period.
 - ⏸ Move to the end – the current date/time.
 - ⚡ Realtime-mode - end-date remains on the current date/time.
 - 🔍 Zoom in
 - 🔍 Zoom out – zoom to 0-100%
 - 👉 Moving mode –click and drag to move the plot area.
 - 📏 Zooming mode – click and drag to zoom.
- Chart area / plot area background and margins
- Navigation scrolling and zooming options
- Pen labels and layout
- Time / value axis and grid lines layouts
- Scaling
 - Auto based on the visible trend-values
 - Fixed fixed value entered
 - Dynamic based on xTagDB properties
- Show / hide trend on alarm limits

9.3.2. Trend Pen Selector

The *'Pen Selector'* control combines the trend pen selecting mechanism, grouping trend pens into sets, and the chart legend in a single control. With the *'Pen Selector'* control, a user can perform in runtime:

- select / deselect tags to shown in on a trend chart from a (filtered) list of tags configured for trending. The tag alias is shown if available, otherwise the tag name is displayed.
- combine selected trend pens into pen sets to easily show a collection of pens. These sets can

be saved and quickly retrieved. Pen sets can be pre-configured and then transferred as a file to the target system ('XLRX\TrendData\PenSets').

- modify the pen properties.

In design mode, you can set properties for the control and the three panes:

- Related Trend chart
 - As an application can have multiple trend charts showing different trends, you need to link the pen selector to a specific trend chart.
- Show / hide *'Tag selection'* pane
- Set filter for tags available
- Show / hide *'Pen set'* pane
- Security level for modifying pen sets
- Allow resize during runtime
- Show / hide *'Pens properties'* pane
- Toolbar buttons
 -  Remove the selected pens from the chart.
 -  Color for the selected pens
 -  Line style for the selected pens.
 -  Marker for the selected pens.
 -  Style for the selected pens.
 -  Chart Style for grouping pens.
 -  Group pens – plot pens in single plot area.
 -  Ungroup pens - separate plot area per pen
 -  Move selected pens up in the list.
 -  Move the selected pens down in the list.
 - Columns to show
- Trend pen columns visibility, position and size.

9.3.3. Trend Navigator

The Trend Navigator is an optional control for data inspection and navigation. It shows the same pens as the Trend Chart but for a longer period. The position of the Chart is displayed in the Navigator and can be moved and resized. The Navigator makes it possible to select a part of the data to view in the Trend Chart.

The Trend Navigator graphical properties are to a large extend like the Trend Chart control. See the section above for those descriptions, this section lists the specific features of the Navigator only

You need to link the trend navigator to a specific trend chart as an application can have multiple trend charts showing different trends. This relation and other properties of the pen selector can be modified by clicking the  button of the control. The main properties are:

- Related Trend chart
As an application can have multiple trend charts, you need to link the trend navigator to a specific trend chart.
- Auto zoom factor
When enabled, the period of the trend navigator and trend chart are linked: when the period of either control changes, the other control period automatically changes according the factor. I.e. a factor of '10' causes the navigator period to be always 10 times the chart period.
When disabled, the period of the trend navigator and trend chart can be changed without affecting each other.

9.4. Advanced trend functions

There are several additional worksheet and VBA functions that can be used for advanced configurations in your application. These functions are described in more detail in the 'Function Reference' help file.

- exTrendReadTag
Reads the raw trend-data for a specific tag over a defined period.
- exTrendReadFile
Reads the trend-data from the specified trend data-file ("xtd").
- exTrendWriteTagToCSV
Writes the trend-data from a specific tag and defined period to a comma separated file.

10. Editing values

This chapter describes how to allow users to enter or modify values (user input) in runtime mode, providing mechanism for checking whether the user is allowed to enter a new value, he has entered a correct value, in the proper format, and within the proper limits.

10.1. Allowing user input

Default all cells on the display sheets are locked so a user cannot enter and type in values during runtime. This is what you want for most cells as otherwise users could start writing all over your display screens. However, for certain values (cells) you want the user to be able to input new values in runtime (user input).

To allow the user to enter values into a cell in runtime, you need to make sure that the input cells are in the UI Range of the 'Worksheet table' and that these are unlocked.

Worksheet	Visible level	Display size	1920 x 1080	UIRange
sStnOvw	0			A1:Q59
sSysOvw	0			A1:Q59
sTotals	0			A1:Q79
sMaint	0			A1:R59
sLegend	0			A1:Q59
sTrends	0			A1:Q59
sAlarms	0			A1:Q59
sEvents	0			A1:R59

Figure 51 Worksheet table UI range

In design mode you can unlock or lock cells from the eXlerate ribbon. When the lock indicator is highlighted , the cell is locked and not modifiable in run-time. When the lock indicator is not highlighted , the cell is unlocked and modifiable in run-time.

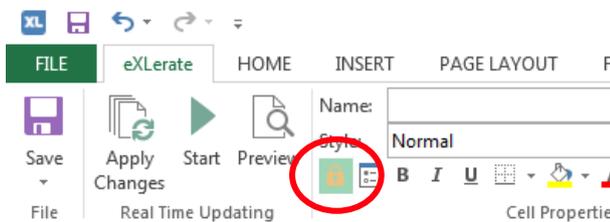


Figure 52 Lock Button on eXlerate Ribbon

The last step is to include the editable cells in the 'xEditing' configuration sheet.

10.2. Editing table

The *xEditing* worksheet contains the Editing Table with the configuration of handling the user input of unlocked cells. It contains the checks for security level, the data type, input limits. When the newly entered value of the editable cell on the display matches all criteria, the new value is written to the target, which can be an internal parameter (calculation tag), alarm limit, or a tag of a connected device.

Figure 53 Editing Table

10.2.1. Configuration

If you are using a recent template application, editing entries for unlocked cells on display sheets may be automatically added when activating the *xEditing* sheet. You always can add entries manually to the sheet.

Each editable cell on a display corresponds to one row on the sheet. You have set-up the entries configuration using the following fields:

- **Class**
Name to group similar editing cells – optional field for structuring your application.
- **Cell**
The address of the editable cell on the display. This can be either a fixed text with a worksheet cell address or an 'exCellProperties' worksheet function to get a property of a cell, in this case the address. The advantage of using the function is that when copying, the reference will automatically be updated.
- **Target**
Reference location where the entered value should be written to upon accepting the value. Multiple targets can be configured by separating them by the ',' (comma) character.

– Target Type

The target location type for the new value:

- | | |
|--------------------------|--------------------------|
| (1) xTargetNone | No target |
| (2) xTargetCell | Target is a cell address |
| (3) xTargetName | Target is a named cell |
| (4) xTargetComm | Target is a comm. tag |
| (5) xTargetAlarmLimit | Target is an alarm limit |
| (6) xTargetAlarmDeadband | Target is a deadband |
| (7) xTargetAlarmDelay | Target is an alarm delay |

– Group

Name for group-wise acceptance of values

See *'Group-wise editing'* below.

– Edit Type

Data type allowed to enter:

- | | |
|------------------|------------------|
| (1) xWholeNumber | Integers |
| (2) xDecimal | Floating points |
| (3) xText | Strings |
| (4) xList | Select from list |
| (5) xDate | Date values ** |
| (6) xTime | Time values ** |

– Edit list

Drop-down list to use – see *'Edit lists'* below.

– Edit Type Alert

Message to show when entered value does not match the required data type. *

Default message is like *"Value does not match the specified datatype"*.

– Min

Minimum value allowed for the new value.

– Max

Maximum value allowed for the new value.

– Min/Max Alert

Message to show when entered value is outside min/max limits. *

Default message is like *"Value should be less/greater or equal to ..."*.

– Enabled

Enables or disables cell editing based.

Enter a Boolean formula.

– Access Level

Minimal security level required to enter a value.

– Confirm msg

(optional) Confirmation text to show when user enters a new value. When entered, the user must confirm before a new value is accepted.

* The following special keywords can be used for alert messages:

- %INPUT% shows the entered value.
- %VALIDATION% shows the exceeded limit
- %FORMAT% date / time format to use

eXlerate Edit functions

On the right side of the configuration section you can find the eXlerate functions that handle the editing in runtime. These functions start with *'exEdit...'* and are described in more detail in the *'Function Reference'* help file.

When adding entries to the sheet, copy the eXlerate functions (in the grey columns) from existing rows as these functions should exist for each editing object.

10.2.2. Group-wise editing

When no group is specified, edited values are accepted, and the target is updated immediately when the user enters a new value in a cell.

When a group is specified, the values are not individually accepted, but only accepted as a group: all targets of the same group are updated simultaneously with the entered values when the user accepts the group.

This mechanism can be used to accept one or more values by clicking on a button. For example, when editing a gas composition, the components should be accepted simultaneously.

When you have entered a group name, you need to create a button for accepting all group values. Add an entry to the button table with an explicit call to the VBA procedure *'AcceptEditGroup_{Group}'*, which will be automatically created when running the Button wizard.

In some cases, it is preferable to perform a check on all the values before accepting them. For instance, verify that the individual components add up to 100% for a gas composition. In this case you need to create your own VBA procedure (macro) to perform this check and accept the values using the *'exAcceptEditGroup("{Group}")'* function. See the *'Function reference'* for details on this function.

10.2.3. Edit lists

With edit lists you can present the user with a drop-down list of values to choose from instead of letting him enter a value.

To create an *'Edit list'*, first define a two-column table. The 1st column contains the value for the target location and the 2nd column contains the corresponding text to display in the drop-down list. Preferably give this table a name.

In the *Editing table* set the *Edit Type* to “lest” and in *Edit list* refer to the table with the list values.

Enter the list as:

- named range “rMyList”;
- unnamed range “xLists!G5:H7”
- list-formatted text “1:GC-A|2:GC-B|3Keypad”
with “:” columns separator
 “|” rows separator

The user can select one of the strings of the 2nd column and the corresponding value of the 1st column will be written to the target.

10.2.4. Date /time editing

By default, a user should enter date/time values in the Windows regional settings format. It is also possible to specify a custom date or time format by setting the 4th argument of the eXlerate *exEditType(...)* function:

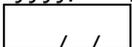
```
fx    =exEditType($M13,$F13,$G13,"yyyy/mm/dd")
```

If this argument is omitted, the Windows Regional Settings format is used.

You can either use a reference to the format as defined in the Style table, e.g. “*xf_Date*”, use the format of the cell itself using the function *exCellProperties({Cell}, xFormat, xAutoRecalc)* or set the format explicitly using the following specifiers:

- yy Year without century (00-99)
- yyyy Year with century (1901 – 2199)
- m, mm Month (1-12 / 01-12)
- d, dd Day of the month, (1-31 / 1-31)
- h, hh Hour in 24-hour format (0-23)
- mm Minute (00-59)
- ss Second as decimal number (00-59)

The user should enter the values in the same format. For instance, when the format for dates is “yyyy/mm/dd”, the following mask is displayed:



10.3. Runtime editing

When a cell is correctly configured for editing, an input box is shown in runtime when a user selects the cell.

Alarm limits		Stream 1	
Temperature	Low alarm	15	°C
Temperature	High alarm	30.00	°C
Pressure	Low alarm	20.000	bar
Pressure	High alarm	60.000	bar

Figure 54 Runtime editing input

When the user enters a new value, it will be checked and if it is a correct value according the editing table. When the input is valid, the new value is accepted. In case the input does not match these settings, a message is shown to alert the user and the value is not accepted.

Alarm limits		Stream 1	
Temperature	Low alarm	Value '150' should be less or equal to '40'	
Temperature	High alarm	150	°C
Pressure	Low alarm	20.000	bar
Pressure	High alarm	60.000	bar

Figure 55 Runtime editing warning

11. Interval periods and events

You can define recurring intervals in your eXlerate applications to perform periodical calculations and to trigger events, like periodic averages, latching totals on certain times, generating periodic reports.

The recurring interval in which certain events take place is called an interval or event. The recurring periods over which the associated calculations take place is called a period.

The two entities highly interact. You need an interval with a defined period to calculate a daily average, and an associated event to print the daily report.

11.1. Interval table

The Interval Table called *'IntervalTable'* is located on the **xTables** sheet. It contains the definitions for interval periods and events. Existing intervals can be changed, and new intervals can be added to this table. You need to run the *Tag & Object Wizard* after making changes to this table.

ID	Name	Type	Count	MM	DD	hh	mm	ss	Periods	StartOn	ResetBy	Previous	Current
1			1					0				0	0
2			1	3				0				33	36
3			1	12				0				24	36
4			2	1				0 0	60			57	58
5	Hour		3	1				0 20	24	6		13	14
6			3	1			30	0				13	14
7	Day		4	1		6	0 30		31	0		18	19
8	Month		6	1	1	6	0 50		12	0		6	7

Figure 56 Interval Table

As with most configuration tables in eXlerate, the table has a header row with field headers, and the data in the table itself. The following fields are available:

- **ID**
Unique index number starting from 1 for the interval.
- **Name**
(optional) unique name for the interval. When a name is entered, it can be used in the application for periodic data and report generation. The tag database will get an associated column with this name prefixed by *'P_'* for defining periodical calculations for tags. For example, the *'Hour'* interval has an associated xTagDB column *'P_Hour'*.
- **Type**
Interval type, being one of the following:

- xSec seconds based
- xMin minutes based
- xHour hourly based
- xDay daily based
- xWeek weekly based
- xMonth monthly based
- xQuarter 3-monthly based
- xYear yearly based

- **Count**
Determines the count of the specified interval types to pass for triggering the event. E.g. a count of 5 with type xMin defines an interval of 5 minutes.
- **MM / DD / hh / mm / ss**
(optional) These define the start (month / day / hour / minute / second) of the interval. i.e. 'hh'='06' 'mm'='00' 'ss'='30', defines interval start at '06:00:30'
- **Periods**
(optional) Number intervals in a range. i.e. '24' periods with type 'xHour' and count '1' creates a 'day' range with 24 hourly intervals.
- **StartOn**
(optional) Starting period for interval range i.e. in the example above, you can define 'StartOn' as '6' for a 'day' range with 24 hourly intervals from 6 am till 6 a.m. (next day).
- **ResetBy**
(optional) Name of another interval that resets the periodical data range. By default, the data range of an interval will be automatically reset when the first event for a new range occurs. i.e. in the example above, all (previous) data is reset at 7 a.m.
- **Previous**
The previous interval number related to the type –set by eXlerate when application runs.
- **Current**
The running interval number related to the type – set by eXlerate when application runs.

11.2. Periodic data

When a name is entered for an interval, it can be used in the application for automatically calculate periodical data. Periodic data is live data gathered and processed during an interval. It can be used to calculate an average process value during a period

or to store totals on the event so these values can be shown on displays and put on reports.

Named intervals create related columns in the Tag Database to enable these automatic calculations and storage. The column names will be the interval name prefixed with “P_”, e.g. an interval named “Hourly” will have a related column named “P_Hourly” in the xTagDB.

Group	ID	TagName	P_Min	P_Hour	P_Day	P_Week	P_Month
	1	S1_SVFR					
	2	S1_MASSFR	WL				
	3	S1_ENFR					
	4	S1_GV_PD		L			
	5	S1_SV_PD					
	6	S1_MASS_PD			L		
	7	S1_EN_PD			L		

Figure 57 Period columns in xTagDB

In these period columns, you can enter a “W”, “L” or both for any tag to indicate that the live data should be averaged during the period (W) and/or be latched at every period change (L). These results are stored and available in the application and can be used to show on displays and reports.

11.2.1. Weighted averages

The average in eXlerate is calculated as a weighted average of a real-time value. The weighted average is like an ordinary average, except that instead of each of the data points contributing equally to the final average, some data points contribute more than others. The contribution of the process parameter to the resulting average depends on the weight factor. This weight factor needs to be specified in the **WeighFactor** column of the *xTagDB*, (without an equal sign). A weight factor needs to be based on an accumulative value, like a totalizer, with which the difference between the previous and the current value is calculated. For a time-weighted average, enter “xNow.Time” into the WeighFactor column.

The equation for a flow-weighted average value is

$$P_{avg} = \frac{\int_{t^0}^{t^n} (P_i * Q_i)}{\int_{t^0}^{t^n} (Q_i)}$$

P_{avg} Weighted average of value P in period $t_0..t_n$

- P_i Current value of P during interval time
- Q_i Weight factor for two consecutive intervals
- t_0 Period start time for calculating the average
- t_n Period end time for calculating the average

Table 6 Weighted Average

Time	Value (P)	WeighFactor	Increment (Q)	P*Q
7:00	15	200	200	3000
8:00	30	210	10	300
9:00	20	0	0	0
10:00	20	220	10	200
11:00	15	320	100	1500
Sum			320	5000

An example of a weighted average is shown in Table 6. The value 20 reported during 9:00 is not used for the averaging process, because the weight factor increment was zero during that hour. The value 15 during 7:00 contributes 20 times as much to the weighted average as the value 10 during 8:00 because the increment is 20 times more. The weighted average is calculated by dividing the sum of the values multiplied by the increments (5000) by the sum of the increments (320). This yields to a weighted average of 15.625 while the arithmetic average is 20.

When you enter for a tag an “W” in a period column, enter a WeighFactor and you run the *Tag & Object wizard*, it automatically creates internal names that will hold the average data so you can use it in your application. The names will be based on the tag name and the interval name.

The following name will be created:

- **x{Tag}.{Interval}.WAvg**
Weighted average value of the running interval.

11.2.2. Latch values

A latched value acts as a sample and hold register. It takes the current value of a tag at a period change and stores that value in memory for the duration of the associated period. When the period elapses, a new value is sampled for the duration of this period.

Latches are especially useful to keep values for a certain period, such as ‘previous day’ or ‘previous

hour' data. Latches are retentive: at system startup, the last stored values are retrieved.

In the figure below, the red line gives the tag live value and the blue line the latched value.

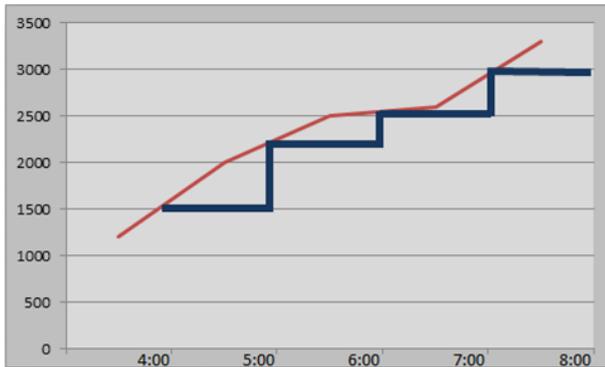


Figure 58 Latched Values

When you enter for a tag an “L” in a period column and you run the *Tag & Object wizard*, it automatically creates internal names that will hold the latched data so you can use it in your application. The names will be based on the tag name and the interval name.

- **x{Tag}.{Interval}.Current**
Current latched value (i.e. NOT the current actual value of the input);
- **x{Tag}.{Interval}.Previous**
Previous latched value;
- **x{Tag}.{Interval}.rPeriods**
Range with all latched values for the number of periods as defined in the interval table. The whole range will be reset when the first interval elapses. I.e. when the interval is defined as hourly interval with 24 periods starting on 6 a.m., the range is reset just before 7 a.m.

The last value, rPeriods, is an array with the number of rows as defined in the interval table periods field. To use this value, enter an Array formula (see section *Excel*).

11.2.3. Latch average values

When you specify both “W” and “L” in the Period column, latches and weighted averages will be automatically combined allowing you to latch the period average on an interval change. It allows you to have a range of period averages, e.g. 24 hourly averages stored at hour changes during a day.

When you run the Tag & Object Wizard, extra tag names become available that contain the combined results:

- **x{Tag}.{Interval}.Current**
Current latched value (i.e. NOT the current actual value of the input);
- **x{Tag}.{Interval}.Previous**
Previous latched value;
- **x{Tag}.{Interval}.rPeriods**
Range with all latched values for the number of periods as defined in the interval table.
- **x{Tag}.{Interval}.WAvg**
Weighted average value of the running interval.
- **x{Tag}.{Interval}.WAvg.{Interval}.Current**
Current latched average value (i.e. NOT the running interval average);
- **x{Tag}.{Interval}.WAvg.{Interval}.Previous**
Previous latched average value.
- **x{Tag}.{Interval}.WAvg.{Interval}.rPeriods**
Range with all latched average values for the number of periods as defined in the interval table.

11.3. Calculation triggers

eXlerate generates so-called calculation triggers for the intervals. These calculation triggers are used by internal functions and may be used for your own application development.

The sequence in which the periodical events as defined in the Interval Table take place is quite important; when a report is to be printed, it is important first to calculate a new result for that interval, then to store the result, after which the report is generated.

Various calculation triggers are defined for each period in the interval table. When an interval event takes place in eXlerate, the following updates take place:

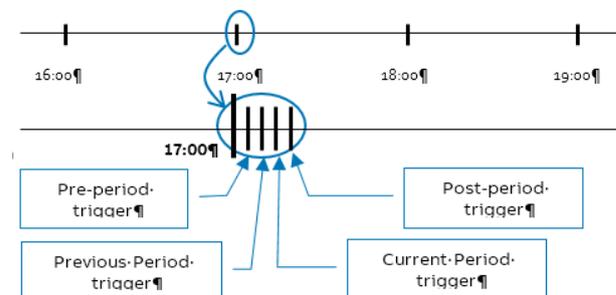


Figure 59 Period triggers

- **xPeriod[...].PreTrigger**
The ‘pre-trigger’ update (eXlerate changes its value) causes the “current” latched value being copied to the “previous” latched value.

- **xPeriod.[...].Previous**
When the previous period field in the Interval Table is updated, it triggers the last calculations for averages of the running interval.
- **xPeriod.[...].Current**
The new value of the current period field in the Interval Table triggers the update “current” latched value with the live / average value.
- **xPeriod.[...].PostTrigger**
The “post-trigger” change will trigger the generation of the reports defined for that period, using the correct values results updated by the previous triggers.

11.4. Reset historical values

The running historical values in an application may be reset using the ‘*Reset Historical Values*’ option from the Tools option in eXlerate. This will reset all data for latches and averages ‘0’ (zero). Trend and log files, or report output files are not touched by this reset.

11.5. VBA events

The application VBA contains several procedures that are triggered by the eXlerate program when certain events occur. These procedures are in the module **modEvents** and may be used to execute other functions and procedures on when these events occur.



You can add your procedures and functions that need be called and executed when certain events, intervals or periods occur.

The VBA events procedures as listed below.

- **OnProjectOpen()**
Actions to perform when the application is opened, either in runtime or design mode. It should at least call ‘*exProjectInit()*’ to initialize the *xlConnect* control.
- **OnEvent()**
Actions to perform each time a certain interval as defined in the **Interval Table** changes. The *ID*, *Name* and *Current* period corresponding to the interval that changing are passed to this *OnEvent()* as the *iEventID*, *strPeriodName* and *iPeriod* arguments.
Note that you need to update these sections in VBA when making changes to the interval table.

- **OnWatchdog()**
Periodically called by the eXlerate to check the VBA dataspace. Unhandled or fatal errors in VBA subroutines may cause a reset. If this is detected, eXlerate calls ‘*OnUserInit()*’ to recover from the reset condition.
- **OnUserInit()**
Actions to perform when a reset of your instance’s VBA global data is detected. It may be used to initialize global constants and settings.
- **OnUserCalculate()**
Actions to perform every calculation cycle.
- **OnBeforeApplyChanges()**
Actions to perform just before applying configuration changes. It may be used to define some settings for communications.
- **OnBeforeActiveMode()**
Actions to perform before communication is started, after the configuration is loaded.
- **OnBeforePassiveMode()**
Actions to perform when the communication is suspended.
- **OnBeforeReport()**
Actions to perform just before a report is generated.
- **OnDaylightSavingChange()**
Called when the daylight-saving time changes.
- **OnBeforeProjectClose()**
Last actions to perform just before the application is closed.



The VBA code that you insert must be fall-through: avoid blocking code like (infinite) loops, modal forms or message boxes etc.



Don’t activate message boxes or ‘modal’ user forms from VBA events procedures as this VBA will halt while the dialog is active.



Also prevent time-consuming code as that decreases the performance and responsiveness of your application.

12. Reports

You can generate and print reports from your eXlerate applications. The reports can be generated automatically on an event like a period change (e.g. every hour or day) or another event, like proving completion or batch end, and on user request.

Company Location						
Daily Report 20/Nov/2018						
Stream 1	Pressure	Temperature	Gross volume	Std. volume	Mass	Energy
	kPa	°C	m ³	Sm ³	t	GJ
7:00	0.	0.00	0.00	0.00	0.00	0.00
8:00	0.	0.00	0.00	0.00	0.00	0.00
9:00	0.	0.00	0.00	0.00	0.00	0.00
10:00	4000.	40.00	7.20	338.40	230.00	12.80
11:00	4000.	40.00	7.20	338.40	230.00	12.80
12:00	4000.	40.00	7.20	338.40	230.00	12.80
13:00	4000.	40.00	7.20	338.40	230.00	12.80
14:00	4000.	40.00	7.20	338.40	230.00	12.80
15:00	4000.	40.00	7.20	338.40	230.00	12.80
16:00	4000.	40.00	7.20	338.40	230.00	12.80
17:00	0.	0.00	0.00	0.00	0.00	0.00
18:00	0.	0.00	0.00	0.00	0.00	0.00
19:00	0.	0.00	0.00	0.00	0.00	0.00
20:00	0.	0.00	0.00	0.00	0.00	0.00
21:00	0.	0.00	0.00	0.00	0.00	0.00
22:00	0.	0.00	0.00	0.00	0.00	0.00
23:00	0.	0.00	0.00	0.00	0.00	0.00
24:00	0.	0.00	0.00	0.00	0.00	0.00
1:00	0.	0.00	0.00	0.00	0.00	0.00
2:00	0.	0.00	0.00	0.00	0.00	0.00
3:00	0.	0.00	0.00	0.00	0.00	0.00
4:00	0.	0.00	0.00	0.00	0.00	0.00
5:00	0.	0.00	0.00	0.00	0.00	0.00
6:00	0.	0.00	0.00	0.00	0.00	0.00

Figure 60 Example Report

12.1. Design vs. runtime

A report in an eXlerate application is a worksheet containing the layout and the references to data (inside the application) to use when the report is generated. The data can be live data, periodical data and calculated values.

The worksheet acts as a template for the actual reports generated in runtime. When opening an application, or when pressing “Apply changes” from the ribbon, eXlerate creates the report template. It will copy the report sheets from the application and save these in a separate template file with all formatting, but without the data on the sheets. The report template files are stored in the “Xlrx\Cache” folder.

The actual reports are stored as separate Excel files on disk in the report path, see ‘*Application Shortcuts*’. Upon generation of the actual report, a snapshot of the actual values is stored. The references and calculated data of the template are replaced by the actual values.

12.2. Report table

All reports must be registered in the **Report Table** on the **xTables** sheet. It contains the definitions for the reports, like the report name, worksheet used as template, the file and sheet name of the report being generated, and the number of initial copies to be printed.

Report	Period	Workbook	Worksheet	Trigger	StartDate	FileName	SheetName	Copies	Options	Password	Printer	Status
Alarm	Hour		Alarm		11/05/2016 10:03	Alarm\Alarm_20190111	09_36_00	0				0
Daily	Hour		Daily		10/12/2018 13:00	Daily\0190111	09_36_00	0				0
Monthly	Day		Monthly		10/12/2018 13:00	Monthly\20190111	Jan 11	0				0

Figure 61 Report table

- **Report**
Internal name for the report
- **Period**
Name of the interval on which the report is automatic generated.
When left empty, reports still can be generated manually or by VBA events.
- **Workbook**
Obsolete – leave empty
- **Trigger**
The period number of the corresponding interval event, at which the report is generated. When Trigger is left empty, or contains an expression yielding to a value < 0, it is generated at every interval event. When the expression at this field yields to a number >=0, it is assumed to be the corresponding period at which the report is to be generated.
- **Worksheet**
Name of the worksheet inside this application to use as template sheet for the report.
- **FileName**
Name of the file to use to save the report when it is generated.
You can use Excel functions to include the date and/or time in the file names so separate files are created for reports generated at different times.
- **SheetName**
Name of the sheet inside the workbook to store the report upon generation.
If a report with the file name already exists, new sheets are added;
Again, you can use excel functions to generate different worksheets.

- **Copies**
Number of copies to send to the printer when report is generated.
When number of copies is set to 0, the report file is still generated and stored on disk.
- **Options**
Additional options (bitwise):
bit 1 (“2”): create both XLSX and PDF reports.
- **Password**
Password for the XLSX report. A user can view the report, but he cannot (accidentally) make modifications without entering the password.
- **Printer**
Printer to use for printing the reports.
When left empty, the printer as specified in the Control Center will be used.
- **StartDate**
Date/time when reports were last generated - automatically update by eXlerate.
- **Status**
Indication if last report generation was successful (=0) or failed (=1).

- Alignment
- Font – (only a single font and text color)
- Borders
- Background
- Charts
You can use standard excel charts on reports. Charts refer to values, and as report values are frozen on generation, the charts should refer to values on the report sheet itself and not to data on other sheets in the application.
- Shape / picture objects
For shapes, pictures or any other object you put in the report sheet, set the “Object positioning” property ‘Move and size with cells’ and enable the ‘Print object’ option.

Prevent using hidden rows or columns on report sheets. Although having hidden rows & columns is fully supported, it can have a negative effect on report file-size and performance.



Save your application and “Apply changes” when you have made changes to report sheets, so report template file is updated.

12.3. Report design

When you design a report worksheet, you may use standard spreadsheet functionality to create the layout and content. This

- Fixed text / values
Type in the values in the cell(s)
- References to application values
Start with an ‘=’ sign and point to the value in your application.
To enter a reference to a range with multiple values (see *Latch values*) select all the cells and enter the reference to the range as an array: press <Ctrl-Shift-Enter> (see *Excel*)
- Formulas
Enter formulas just like regular excel formulas. When a report is generated, all formulas are replaced by values. When you want to have formulas on the generated reports, you need to define the formula in between quotes so it is considered as text.
For example, to display the current date/time, use the syntax “=“NOW()”” as this formula is converted into “=NOW()” when the report is generated. As a result, it will display the date / time of the re-print or preview.
- Set cell formatting:
 - Style (numerical format)

12.4. Report generation

In design mode, the reports can be manually generated from the eXlerate ribbon to test the layout and functionality.

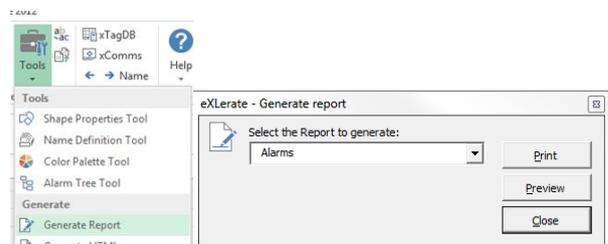


Figure 62 eXlerate - Generate report tool

In runtime, the reports are created on events as defined in the Report Table, VBA code or by a user action like pressing a button that you have defined in your application.

Report generation consists out of the following steps:

- Create a new file / sheet based on the report template and the file / sheet name of the Report table.
- Create snapshot of the data on the report sheet from the application.
- Put the snapshot data in the report file / sheet

- When report preview is set: Open a temporary working copy for preview in excel
- When report number of print copies is set: Open a temporary working copy in excel and let excel print the sheet.
- When PDF report option is set: Open a temporary working copy for creating a PDF from the excel file.

12.5. VBA report functions

In VBA functions are available with which you can programmatically generate or reprint reports. These functions are described in more detail in the 'Function Reference' help file.

- `exGenerateReport()`
Generate a report from the Report table
- `exReprintReport()`
Reprint an existing report file / sheet.
- `exPrintCurrent(...)`
Prints the current worksheet on the report printer.

13. Redundancy

Spirit^{IT} eXlerate support different types of redundancy to allow a system to continue functionally running when a single point of failure occurs. This chapter describes the different redundancy configuration types and how to configure these within an eXlerate application.

Within eXlerate the different redundancy types can be combined to achieve a higher level of availability.

13.1. Redundant communication links

Redundant communications consist of having two (network/serial) communication links to a single device. The data originates from a single source (device) and is transferred over two different physical links. If one link fails, the data is still available using the second link.

Redundant communication in an eXlerate application requires 2 communication links (protocols). You have two different methods, described below.



Figure 63 Redundant communication link

13.1.1. Full redundant communication

All data will be transferred using both links. Both links will have the same set of queries and data points. The data set to use can be selected from one of these sets.

The advantage of this method is that the setup is straightforward – communication links are configured as separate devices and the related queries – and the switch-over is bumpless, without re-establishing the communication links. The disadvantage is that it puts more strain on

the server and the device because it effectively doubles the number of communication points.



For full redundant communication, configure your application as follows:

- Two protocols - one for each link (e.g. “192.168.1.100” and “192.168.2.100”)
- Two sets of queries - one set for each protocol. These sets will be identical for both protocols with either in the same xTagDB column or two different xTagDB columns assigned.
- Tag database with one set of tags, set-up with the following columns:
 - Value the value used in the application; the values are set by both query sets, or a calculation selects the value from the (optional) link value columns as defined below, based on protocol status.
 - Link1 (optional) values from link 1
 - Link2 (optional) values from link 2
 - Query query IDs of both devices, separated by comma, e.g. “5, 13”
 - Query1 (optional) query ID for link 1
 - Query2 (optional) query ID for link 2
 - Update1 write function for 1st link (Query1)
 - Update2 write function for 2nd link (Query2)
- Calculate the link to use (optional), like `=IF (NOT (xFX1_LAN1.SAlarm.Raised), 1, 2)`

13.1.2. Validity check redundancy

The data will be transferred using one link. The redundant link is only used for checking the availability of the line. When the primary link fails, the communication is swapped: the 1st protocol for the data communication switches to the link of the 2nd protocol; the 2nd protocol to monitor the status of the link switches to the link of the 1st protocol.

The advantage is that this setup puts much less strain on the system and device while providing the same level of redundancy. The disadvantage is that if the switch-over is not bumpless as it requires re-establishing the communication when swapping the links and the set-up is more complex.



For redundancy with validity check, configure your application as follows:

- Two protocols - one for each link
(e.g. “192.168.1.100” and “192.168.2.100”)
- Enable that the protocols can swapped, using eXlerate “*exSetAlternateDevice()*” function:


```
exSetAlternateDevice (ID1, 2, Dev2, xAutoRecalc)
exSetAlternateDevice (ID2, 2, Dev1, xAutoRecalc)
```
- Two different sets of queries:
 - 1st set contains full data set
 - 2nd set contains only one query for checking the availability of the line
- Tag database with one set of device tags, set-up with the following columns:
 - Value the value used in the application, related to the 1st set of queries
 - Query ID numbers of the 1st set
 - Update write function for 1st set
- Tag database with one additional tag for the validity check of the redundant link:
 - Value validity check value related to the query of the 2nd link
 - Query ID number of the 2nd link query
- Calculate the link to use, with a time-out when switching links on a communication link failure. It requires 3 calculations:
 - Calc.Link Link to use - set by 2nd calc.
 - Calc.SetLink Update link to use based on protocol status, with timeout of 15 sec:


```
exCopy (IF (STS1<>1, 1, 2), Calc.Link, AND (STS1<>4, xNow.Time>(Calc.SwitchDT+15/86400)) )
```
 - Calc.SwitchDT Time of last link change:


```
=exNow (Calc.Link)
```
- Switch the links for the protocol using eXlerate “*exSelectDevice()*” function for both protocols, based on the calculated value:


```
exSelectDevice (ID1, Calc.Link, xAutoRecalc)
exSelectDevice (ID2, Calc.Link, xAutoRecalc)
```

13.2. Redundant devices

Redundant devices consist of having two devices performing the same task (measurement). These devices might or might not be synchronized by the devices themselves. Both devices communicate with the eXlerate application.

Redundant devices in an eXlerate application requires 2 communication links (protocols). You need to two different methods, described below.

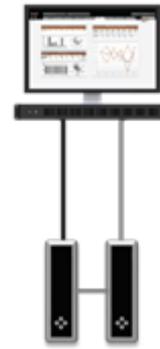


Figure 64 Redundant device

The data originates from a two sources (devices).

13.2.1. Separate device tags

With separate tags per device, the data of both devices is handled independently: the data of each device will be available for alarming, trending, averaging, latching, etc.

This is the simplest configuration as each device is handled separately. The disadvantage is that it requires the double amount of tags.



For separate device tags, configure your application as follows:

- Two protocols - one for each device
- Two sets of queries - one set for each protocol. These sets will be identical for both protocols with a different tag set (row numbers) in the xTagDB assigned.
- Tag database with two sets of tags – one for each device - with the following columns:
 - Value the value used in the application, for the related device (protocol / queries)
 - Query ID numbers of the related query
 - Update write function for the query

13.2.2. Combined device tags

You can combine the data from the redundant devices into a single set of tags. Each tag will contain the values of both devices and based on a status calculation, one of the sets of values will be used for alarming, trending, averaging, latching, etc. The device of which the data is being used, is normally called the “Duty” or “Main”, and the other device is called “Backup” or “Standby”. The device that will be the “Duty” can be determined by the devices itself or you can use eXlerate to determine the “Duty” based on (communication) status.

 For combined device tags, configure your application as follows:

- Two protocols - one for each device
- Two sets of queries - one set for each protocol. These sets will be identical for both protocols, with different columns in the xTagDB assigned.
- Tag database with one set of tags for the combined devices, with the following columns:
 - Value in-use value (Duty device)
 - Value2 values from device 1
 - Value3 values from device 2
 - Query query IDs of both devices, separated by comma, e.g. "5, 13"
 - Query1 (optional) query ID for device 1
 - Query2 (optional) query ID for device 2
 - Update1 write function for 1st device
 - Update2 write function for 2nd device

- Calculate the "Duty" device, like

```
=IF (AND (NOT (xFX1_LAN1_STS.SAlarm.Raised),
        xFX2_LAN1_STS.SAlarm.Raised), 1,
     IF (AND (xFX1_LAN1_STS.SAlarm.Raised,
             NOT (xFX2_LAN1_STS.SAlarm.Raised)), 2,
        xFX1_Duty_Sts.Value2))
```

13.3. Redundant supervisory

A supervisory system of multiple computers with the same functionality protects against a single failure and allows operation control on different locations. An eXlerate project configuration allows this functionality by the Server & Client network configuration.



Figure 65 Redundant servers

With eXlerate, one application serves both server and client roles. It is not necessary to create separate applications; the same application can be used on all computers in the supervisory system.

13.3.1. Servers

For the servers, a "Duty" / "Standby" concept is used. The set-up has only one "duty" server, which performs the actual communication, maintain the database and generates reports and alarms. The "Standby" server(s) continuously synchronize all the data, such as reports, averages, and database, from the duty server.

When the "Duty" server fails, a "Standby" server will automatically take over and will be promoted to "Duty" server. Data consistency is ensured as it was continuously synchronized with the previous "Duty server. When the failed server comes back online, it will synchronize itself with the duty server and take its role as a standby server.

13.3.2. Clients

A client visualizes the data and allows user interactions, but it doesn't communicate directly with devices, nor generates alarms nor reports. A client it reads the data from the Duty server.

Client computers can be either dedicated client or non-dedicated client. A dedicated client is explicitly configured in the application and can be monitored by others. A non-dedicated client is not configured in the application and cannot be monitored. It still can connect to the servers and participate. A non-dedicated client can for instance, be a service- or engineering laptop which is only used during maintenance.

13.3.3. Configuration

An eXlerate application for a Client / server system requires the 'xNet' sheet. It is a ready-to-use configuration sheet and it can be found in the 'MyNet' sample application.

 To implement client / server functionality, open the 'MyNet.xlsx' as a second workbook in your eXlerate application.

The 'xNet' sheet and related functionality will be automatically inserted into your application.

Project Configuration						
	Name	IP1	IP2	IP3	IP4	AllowDuty
Server 1	SERVER01	192.168.1.1	192.168.2.1			TRUE
Server 2	SERVER02	192.168.1.2	192.168.2.2			TRUE
Server 3						TRUE
Server 4						TRUE
Client 1	CLIENT01	192.168.1.11	192.168.2.11			
Client 2	CLIENT02	192.168.1.12	192.168.2.12			
Client 3						
Client 4						
Client 5						
Client 6						
Client 7						
Client 8						

Figure 66 xNet Client / server configuration sheet

The standard 'xNet' sheet allows you to configure up to 4 Servers, 8 Clients - up to 4 network addresses each – and 4 printers.



Set up the computer names and IP addresses once you have the 'xNet' sheet added in your application.

- Name Computer name as configured in Windows.
- IP# (optional) IP addresses by which the computer can be accessed.
It is highly recommended to explicitly specify IP addresses. Accessing computers on a network merely by using their computer-name is not guaranteed to work for systems that don't use name-servers such as WINS or DNS.
- Allow Duty Specifies if a server can become the Duty Server.
Only Duty Servers communicate with devices. If this is set to FALSE, the Server cannot become duty, but it will have the synchronized databases and reports locally, in contrast with clients that retrieve the data from the Duty server and not retain it locally.
- Type Specifies the printer as a Report printer or as an Events printer.



For clients, set-up drive sharing and mapping as the client computers don't have the reports files locally:

- On the servers, create a file-share for the eXlerate report folders;
- On the clients, create a network drive-mapping for each server to these shared folders;
- On the clients, set the report folder to these mapped drives in the Control Center shortcut: If multiple servers are used, each server should drive-mapping should be separated using a '|' character as shown below. The order is as "Server1\Reports | Server2\Reports".

Report output path X:\Reports|Y:\Reports

The 'xNet' worksheet contains a separate section for advanced settings. These settings allow you to tweak the communication between servers and clients. The default values should be used in most normal situations.

Setting	Default	Description
Port	9666	TCP/IP port used for communicating with other

Setting	Default	Description
		clients and servers.
Update Interval	3	Cycle time in seconds for Network calculations.
Switchover period	10	Time-out when switching over (min. 3 times update interval).
Local Override	0	When a value is set, the computer is forced in a fixed duty selection, e.g. when set to '2' the 2nd server will be forced to be the duty server.
Time synchronization Hour	3	Hour for periodic time synchronization between servers and clients. If empty, time synchronization is performed every hour or not at all, depending on the 'Minute' setting.
Time synchronization Minute	30	Minute of the hour for periodic time synchronization between servers and clients. If left empty, no time synchronization will be performed.
Max. Print Jobs	5	Maximum number of print jobs allowed for the printers.

13.3.4. Application development

The 'xNet' sheet provides an elaborate set of names which can be used in your application to show the status on displays, animations, generate alarms, etc.

Names

The following global names are available:

Table 7 Network status names

Name	Description
Net.Configured	Indication that the license is an eXlerate Client / server edition
Net.Started	Indication the network is configured, and communication is started
Net.StartTime	Time when the communications were started
Net.UpdateTime	Time of last calculation cycle

Name	Description
	for network functions
Net.WaitTimeExpired	Checks if the system is still waiting for Duty Server determination after communication has started. The wait time is different for each server and is based on the Switchover time and the Server ID
Net.Local.ServerID	ID of the local server. '0' if it is not a server.
Net.Local.ClientID	ID of the local client. '0' if it is not a client.
Net.Local.IsClient	TRUE when the local computer is a client.
Net.Local.IsDutyServer	TRUE when the local computer is the duty server.
Net.Duty.ID	ID of the duty server. '0' if no duty is selected.
Net.Duty.Name	Computer name of the duty server. 'None' if no duty is selected.
Net.Duty.SwitchOver.Timer	Timer counting when Duty switch-over occurs to prevent 2 nd switch-over occurring within the Switchover period

When set to *'Single cells names'*, the following names are available:

Table 8 Network cell naming

Name	Description
Net.Server{x}.Name	Computer name of the Server {x}, with {x} being the server number.
Net.Server{x}.IP1 Net.Server{x}.IP2 Net.Server{x}.IP3 Net.Server{x}.IP4	IP addresses for Server {x} .
Net.Server{x}.Status	Overall connection status for server {x}, as a number between 0 and 4
Net.Server{x}.Status1 Net.Server{x}.Status2 Net.Server{x}.Status3 Net.Server{x}.Status4	Connection status for a specific IP address (network adapter).

Name	Description
Net.Server{x}.Duty	Duty server ID selected by that particular Server; '0' if no duty is selected.
Net.Client{x}.Name	Computer name of the Client {x}, with {x} being the Client number.
Net.Client{x}.IP1 Net.Client{x}.IP2 Net.Client{x}.IP3 Net.Client{x}.IP4	IP addresses for Client {x} .
Net.Client{x}.Status	Overall connection status for Client {x}, as a number between 0 and 4
Net.Client{x}.Status1 Net.Client{x}.Status2 Net.Client{x}.Status3 Net.Client{x}.Status4	Connection status for a specific IP address (network adapter).
Net.Client{x}.Duty	Duty server ID selected by that Client; '0' if no duty is selected.
Net.Printer{x}.Name	Windows name of the Printer {x}, with {x} being the printer number
Net.Printer{x}.IP1 Net.Printer{x}.IP2 Net.Printer{x}.IP3 Net.Printer{x}.IP4	IP addresses for Printer {x} .
Net.Printer{x}.Status	Overall communication status of the Printer {x}, as a number between 0 and 4
Net.Printer{x}.Status1 Net.Printer{x}.Status2 Net.Printer{x}.Status3 Net.Printer{x}.Status4	Communication status of a specific IP address (network adapter).

When using the single cell names, you can directly refer to the individual cell values. For example, if you want to refer to the overall status of server 2, use a formula like:

=Net.Server2.Status

When set to *'Range names'*, the following names are available:

Table 9 Network named ranges

Name	Description
Net.Servers	Range with the Computer names and IP address of the Servers
Net.Servers.Status	Range with the communication

Name	Description
	status of the Servers, as a number
Net.Servers.StatusTxt	Range with the communication status of the Servers, as text
Net.Servers.Duty	Range with the IDs of the duty server as selected by the individual servers;
Net.Servers.DutyName	Range with the names of the duty server as selected by the individual server;
Net.Clients	Range with the Computer names and IP address of the Clients
Net.Clients.Status	Range with the communication status of the Clients, as a number
Net.Clients.StatusTxt	Range with the communication status of the Clients, as text
Net.Clients.Duty	Range with the IDs of the duty server as selected by the individual clients;
Net.Clients.DutyName	Range with the names of the duty server as selected by the individual clients
Net.Printers	Range with the Computer names and IP address of the Printers
Net.Printers.Status	Range with the status of the Printers, as a number.
Net.Printers.StatusTxt	Range with the status of the Printers, as text.

When using the range names, you can refer to the individual cell values using the INDEX function. For example, if you want to refer to the overall status of server 2, use a formula like:

```
=INDEX(Net.Servers.Status,2,1)
```

Servers / clients connections can have the following status:

0. Running
The application can connect and communicate with eXlerate application on the specified name / IP address.
1. No heartbeat
The name / IP address is valid, but it can't connect.

2. Not connected
The name / IP address is not available on the network.
3. Not started
eXlerate communication has not been started
4. -
No computer name / IP address configured

Printer connections can have the following status:

0. Ready
Printer is available and ready for printing
1. Busy
Printer is printing
2. Warning
Printer is available, but returns an error code
3. Not available
The computer cannot connect to the printer
4. -
No printer / IP address configured

Pre-defined animations

The 'xNet' sheet contains pre-defined animation objects for the status of the servers, clients and printers. When you use shapes with these names on your displays, these will be automatically animated based on these statuses.

Table 10 Network animations

Name	Description
Server{x}	Main name of the (grouped) object for server {x}, with {x} being the server ID. Set visible when the server name is set.
Server{x}.Link1	Sub-shape to show a specific link.
Server{x}.Link2	Set visible when an IP address is set.
Server{x}.Link3	
Server{x}.Link4	
Server{x}.Flt1	Status indication of a specific link.
Server{x}.Flt2	Set visible when communication fails.
Server{x}.Flt3	
Server{x}.Flt4	
Client{x}	Main name of the (grouped) object for client {x}, with {x} being the client ID. Set visible when the client name is set.
Client{x}.Link1	Sub-shape to show a specific link.
Client{x}.Link2	Set visible when an IP address is set.
Client{x}.Link3	
Client{x}.Link4	
Client{x}.Flt1	Status indication of a specific link.
Client{x}.Flt2	Set visible when communication fails.
Client{x}.Flt3	

Name	Description
Client{x}.Flt4	
Printer{x}	Main name of the (grouped) object for printer {x}, with {x} being the printer ID Set visible when printer name is set.
Printer{x}.Flt	Status indication of a specific link. Set visible when communication fails.
Printer{x}.Status	Status indication of a printer. RED printer reports an error YELLOW printer reports a warning invisible printer status normal
Printer{x}.Busy	Printer is busy indication. Visible when printer reports it is busy.

Parameter synchronization

The *exStoreValue()* function serves a dual role in client / server systems. It responsible for retentive storage of parameters and synchronizes parameters between clients and servers.

Parameters are initialized on startup with the last stored value on the Duty server. When a parameter value is changed on any server or client, the parameter value on all other servers / clients is automatically updated and stored.

It is essential that the 'Trigger' argument of the *exStoreValue()* function contains the value 'xAutoRecalc'. for correct initialization.

Shared values

Shared values are synchronized on all servers and clients in a system. You need to specify the condition when the value should be written and when it should be read. When compared to synchronized parameters, they have the following different characteristics:

- Synchronized parameters are always written;
- Shared values are only written when the write-condition is 'True', else they are read.
- Synchronized parameters are retentive
- Shared values are not retentive.

Conditional functionality

As all VBA code is executed on all servers and clients, you need to keep in mind that only the duty server should perform specific tasks, such as generating automatic reports, controlling valves, etc. During development you should prevent

certain functionality from running on any other computer but the duty server.

 Use the following functions to implement conditional functionality:

- *exIsClient()*
Checks if the computer is a client.
- *exIsDutyServer()*
Checks if the computer is the duty server.

Writing to devices is also possible from all servers and clients, using the functions as described in *'Update device values'*. When a value is changed on either computer using these functions, it causes to write the data. To restrict the writing, see the functionality described in section *'Advanced Update mode'*.

13.4. Communication routing

By default, the Duty server performs the communication with connected devices. All other servers and clients read and write data through the Duty server. When the Duty server loses the communication with a device, a switch-over would be required to restore communication, but only if the other server has no communication problem. If both servers have a communication problem, the system would be crippled. If both servers have the  same problem, there is no way to work around this problem. When the servers have different problems - Server 1 can't communicate with device 1; Server 2 can't communicate with device 2 - the system availability is not necessarily in danger: eXLerate can re-route communication from the Duty server to another server.

It is possible to re-route communication on query level, as multiple devices can be attached to a single protocol (multi-drop systems).

 To setup communication re-routing, configure the following functions:

- Enable the communication protocols on the required servers with the *"exSelectDevice()"* function.
- Monitor the status of the queries on the servers using the *"exQueryStatus()"* function.
- Create a calculation for implementing the routing behavior.
For example, use the duty server when the query status on that server is okay, else the standby server:

```
IF(CHOOSE(Net.Duty.ID,0,QuerySts1,QuerySts2)  
=0,Net.Duty.ID,IF(Net.Duty.ID=1,2,1))
```

- Re-route the query to a specific server with the *“exQueryServer()”* function.

When the *“exQueryServer()”* function is omitted, eXLerate uses the Duty server for the communications. This is the default behavior.

14. Databases

A database is a structured collection of data. It may be anything from a simple shopping list to a picture gallery or the vast amounts of information in a corporate network. A relational database stores data in separate tables. This fits nicely in the spreadsheet concept of rows (records) and columns (fields).

In typical eXLerate applications, databases are used for storing events and historical data of the connected measurement and control devices.

A great feature of eXLerate is redundancy support for its system database. The system database tables are automatically synchronized between multiple servers without any programming effort.

14.1. The system database

eXLerate has a system database (previously known as the embedded database) for event logging and to synchronize data between redundant servers.

Since version 4.2, eXLerate supports two options for the system database:

- the embedded system database
- the external system database.

The system database is identified with ID '0'.

To achieve a solid redundancy implementation, the system database is automatically synchronized on redundant systems. Two types of synchronization can be distinguished:

- **Initial synchronization**
Upon startup, a server determines whether it is the duty server or not. If not, it will copy the complete database from the duty server over its own database and show the 'Synchronization' window. When completed, both systems databases will be identical.
- **Run-time synchronization.**
Once the databases are completely synchronized, they are constantly kept up to date.

This mechanism ensures all servers always have an up-to-date database. Failure of a single server will not cause any data-loss.

14.1.1. The embedded system database

The embedded system database does not require any additional database server. It will use the embedded MySQL database engine.

The embedded system database is not accessible externally.

Configuration is easy.



Specify the database path in the application shortcut of the Control Center.



Figure 67 Embedded database path

When you have a system with more than one eXLerate application, the database path for each application can be configured to be the same path or a different path.

At startup, eXLerate will check the database. It will create the database and the required tables automatically when it doesn't exist. If the database is corrupt, it will try to repair the it.

The embedded system database can be fine-tuned by changing the default settings in the application, using the `exSetDatabaseProperty()` function. This allows you to set the number of concurrent connections allowed. Default value is 5 concurrent connections.

14.1.2. The external system database

Since version 4.2, eXLerate supports the use of a local database server instead of the embedded database for the system events database. The benefit of a local database server is that it can be accessed by any SQL Client e.g. for troubleshooting.

The external database server used for the system database must be MySQL compatible and must run on the same computer where eXLerate runs. Furthermore, the storage engine must be 'MyISAM' for all tables.

At ABB Spirit^{IT} we have used and tested this functionality on the MariaDB database server. Other MySQL compatible database servers may work as well but are not tested.



Configured the external system database in the application shortcut of the Control Center.

Figure 68 External database path

- Server address
This must be the local PC: 'localhost'.
- Server port
Database server port to use to communicate - Typically port 3306.
- Username
User to connect to the database server.
- User password
Password of the user to connect to the database server.
- Database
Name of the database to access.

14.1.3. System database tables

When started, the system database contains the following tables. These tables are required for eXLERate to function:

- COMMAND_LOG
Table used for data synchronization between redundant servers.
- EVENTS
Table used for storing alarm transitions, system events and application specific events. It contains the following fields:

Table 11 Event table fields

Column	Data Type	Description
ID	BIGINT	Unique ID of the event
DATETIME	DATETIME	Date and time the event was stored
CLASS	VARCHAR	Event class (e.g. 'Alarms', 'Security', 'Parameter')
TYPE	VARCHAR	Event type (e.g. 'Ack', 'Logoff', 'Change')
LOCATION	VARCHAR	Location (e.g. 'FX01',

Column	Data Type	Description
		'SVC1')
USER	VARCHAR	Username or 'System'
MESSAGE	VARCHAR	Event message
EXTRA1	VARCHAR	Additional event field 1
EXTRA2	VARCHAR	Additional event field 2

The primary-key is defined for column 'ID', so the table will never have multiple records with the same 'ID'. Furthermore, an index is defined on column 'DATETIME'. This index ensures the fast operation of queries when sorting records by date-time.

14.2. Foreign databases

eXLERate applications may connect to one or more additional databases, referred to as foreign databases. Foreign databases are not synchronized.



To configure foreign databases, use the functions:

- **exConfigureDatabase()**
Creates a connection to the foreign database and assigns a unique database ID for internal reference. The following database drivers are supported:
 - MySQL MySQL database
 - SqlServer Microsoft SQL Server
 - OleDb Generic OLEDB driver
- **exSetDatabaseProperty()**
Sets a property of a database driver to a new value. The supported properties are driver specific.
 - Host
Address of the computer where the database is located. This can be either the computer name or an IP-address.
 - Port
Port that is used to communicate with the database.
 - Database
Name of the database to access.
 - User
Username to use to login to the database
 - Password
Password to use to login to the database.

- Concurrent_connections
Maximum number of simultaneous connections to access the database. Increasing this value may improve the ability to process more SQL statements in parallel.
- **exSQLLastDriverSpecificInfo()**
Obtains driver specific information returned by the last executed SQL statement.
 - MySql_error MySQL error code.
 - MySql_time Time in seconds it took to execute the SQL statement.
 - OleDb_error SQL/OLEDB error code.
 - OleDb_time Time in seconds it took to execute the SQL statement.
 - Thread Thread ID of the SQL statement.
 - Affected_rows Number of rows that have been affected.
- **GetDriverSpecificInfo**
VBA equivalent of exSQLLastDriverSpecificInfo

The setup and configuration of 3rd party database platforms is not covered by this document.

14.3. Application databases

In eXlerate application the databases are identified by unique database IDs. The ID '0' always refers to the system database. Values '1' or higher refer to foreign databases configured with 'exConfigureDatabase()' function.

If an application needs to access the local database regardless if it is the duty or standby server, then you can use the database identifier "local_embedded". It is not recommended to access the "local_embedded" database as changes to the local database are not automatically synchronized.

14.3.1. User definable tables

The eXlerate databases can be extended with user definable tables. For the system database tables, the 'MyISAM' storage engine must be used. 'MyISAM' tables consist of these three files:

- <Table>.FRM MyISAM Definition File
- <Table>.MYD MyISAM Data File
- <Table>.MYI MyISAM Index File

During startup of eXlerate, the Event Logger will show an event for every user defined table found in the system database.

User definable tables in the system database are automatically synchronized. Upon startup each server should contain the user defined tables with the same layout. After synchronization, the data in the tables on all servers will be identical.

When manually copying table files from one database server to another, make sure that both database servers are stopped. In case of the eXlerate system database, make sure the application is completely shut down.

14.3.2. SQL queries on worksheet

A worksheet is highly suitable for representation of a relational database, where data is organized in records and fields. The SQL interface in eXlerate can be fully configured from worksheets. Using the worksheet functions, these records and fields directly translate into rows and columns, respectively.

Once a database is configured – eXlerate system database on starting the application or foreign databases with 'exConfigureDatabase()' function - it can be accessed using so-called 'SQL queries': an SQL statement executed on a database.

To implement SQL queries from the



Use the following worksheet functions to implement SQL queries on worksheets within an application:

- **exSQLCreateQuery()**
Create a query for the specified database. A unique query ID must be provided which need to be used in subsequent database calls.
- **exSQLExecQuery()**
Execute the query with the ID provided by *exSQLCreateQuery()* and return any result set of the query to the worksheet in rows and columns. This is the 'workhorse' function executing SQL statements.



Minimize the number of times SQL queries are executed as these can take a lot of time and produce large result sets.

The functions below allow you to analyze the results of an SQL query that is executed and process the result in your application.

- **exSQLRowCount()**
The number of rows of the data returned for the SQL query.

- **exSQLColumnCount()**
The number of columns of the data returned for the SQL query.
- **exSQLLastDurationTime()**
The time it took to execute the SQL query.
- **exSQLLastError()**
The error number returned when the SQL query was executed.
- **exSQLDiagnosticalValue()**
A diagnostical value for the SQL query, e.g. how many times a query (with a specific command) has been executed.

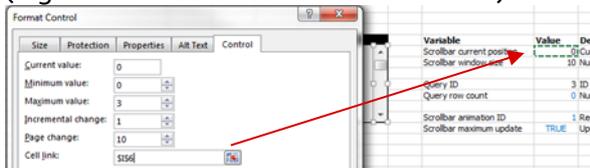
 To view the data retrieved from a database table (with an SQL *'Select'* query), use:

- **exSQLField()**
A single field of the resulting data set for an SQL query.
- **exSQLViewQuery()**
The resulting data set for an SQL query in a 2-dimensional array, or a partial selection of the data set.

 Use scrollbars to display large data sets. By using scrollbars, you can dynamically change the start row and/or columns:

- Select the range where the data should appear
- Enter the *exSQLViewQuery()* formula (enter with <Ctrl-Shift-Enter> as it is an array function)
- Create a scrollbar next to the data with a logical name.
- Link the scrollbar current position to the cell referred in the *exSQLViewQuery()* formula as start row / column.

(Right-click and select 'Format Control...')



- Update the scrollbar boundaries as queries can return result sets with changing row counts:
 - Create animation object:
=exShapeID(<Scrollbar name>,xAutoRecalc)
 - Update boundaries with function:
=exShapeMinMax(<ShapeID>,<Min>,<Max>)
The minimum value should remain '0'.
The maximum value is calculated as follows:
<max> = <row count> - <scroll window size>

14.3.3. SQL queries in VBA

The main interface in VBA to access the database is the *'SQLCmd'* object. It must be created explicitly in your VBA code before it can be used.

```
Dim oSQL As New SQLCmd
```

 Use the following VBA functions to execute SQL queries within VBA:

- **Execute()**
Executes a SQL statement on a database and waits for it to complete or time-out. Statements are executed in series and will block the application while waiting to finish.



- **ExecuteAsync()**
Executes a SQL statement on a database and returns immediately. Statements defined in a batch are executed in parallel and don't block the application.



When an SQL statement is executed, the results are stored inside the *'SQLCmd'* object. These results can be accessed using the other properties and functions of the object.

- **RowCount**
The number of rows of the data returned for the SQL statement.
- **ColumnCount**
The number of columns of the data returned for the SQL statement.
- **Time**
The time it took to execute the SQL query.
- **ErrorCode**
The error number returned when the SQL query was executed
- **ErrorDescription**
The error description returned when the SQL query was executed

- **GetDriverSpecificInfo()**
Driver specific information field that was returned by the last executed SQL statement.
- **Field()**
A single field of the resulting data set for an SQL statement.
- **GetArray()**
The resulting data set for an SQL statement in a 2-dimensional array.
- **IsAsyncInProgress()**
Checks whether an asynchronous SQL statement is still in progress.
- **WaitForAsyncAsync()**
Waits for an asynchronous statement to complete or time-out.
- **CancelAsync()**
Cancels an in progress asynchronous statement.

To make it easier for to execute and check the progress of SQL statements in parallel, the *'SQLCmdBatch'* object is available. *'SQLCmd'* objects can be added to a batch after which the progress of the whole batch can be checked.

The following functions are available for the *'SQLCmdBatch'* object:

- **Add()**
Adds an *'SQLCmd'* object to a batch.
- **IsAsyncInProgress()**
Checks whether any of the asynchronous statements is still in progress.
- **WaitForAsync()**
Waits for all asynchronous execute statement to complete or time-out.
- **CancelAsync()**
Cancels all in progress asynchronous statements.

15. Terminal Services

Terminal Services offers the ability to view and manage an eXlerate system remotely using one or more remote desktop sessions (RDP). With Remote Desktop, the eXlerate program runs on one system, while being displayed on a separate computer. This gives the benefit that you do not need to install any software on your computer in order to view or manage the eXlerate system.

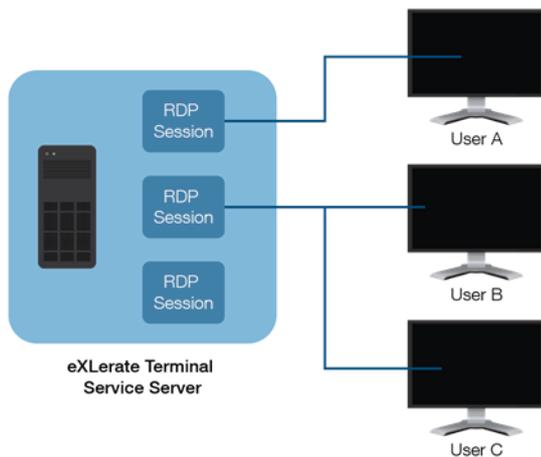


Figure 69 Terminal services

15.1. Requirements and licenses

In order to use the eXlerate Terminal Services, the following software and licenses are required.

- Windows 10 supports 1 simultaneous remote desktop connection.
- Windows Server with the Terminal Services Role enabled and the appropriate licenses (TS CAL's) allows running multiple remote desktop instances simultaneously.
- Microsoft Office with Volume License.
- eXlerate Terminal Services license

15.2. Configuration

Using eXlerate in combination with Terminal Services requires configuration at both the Operating System level and the eXlerate level.



Setup Microsoft Windows to allow remote desktop sessions:

- Install 'Remote Desktop Services' role on Windows Server.
- Purchase and install 'RDS CALs' for the users or computers which are to connect to the server.
- Set the remote desktop services setting 'Restrict each user to a single session' to ".

It is outside the scope of this document to describe these steps in detail. Consult the Internet if you need information on a topic.

In eXlerate the Terminal services need to be set-up to allow running multiple user sessions simultaneously.



Select the '*Enable Terminal Services Mode*' in the Control Center.

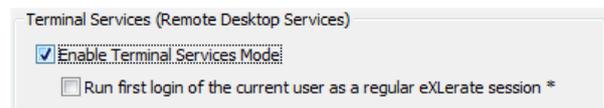


Figure 70 eXlerate Terminal services

With this option enabled, eXlerate applications started on the system run in 'Terminal Services Client Mode'. It allows a single application to be started multiple times on the same computer, once in each terminal services session.

You may run a separate computer for Terminal Services Clients, but you may also combine an eXlerate server, or a standalone system with Terminal Services support. This requires that your first launched session runs as a regular eXlerate session (i.e. eXlerate Server Mode or eXlerate Standalone Mode).



Enable '*Run first login of the current user as a regular eXlerate session*' to start the 1st session as eXlerate Server or Standalone.

To use Terminal Services, your application needs to contain client / server support: it requires a 'xNet'sheet and the server table should contain the names and IP-addresses of the server(s). See section '*Redundant supervisory*' for more information on configuring servers / clients.

15.3. Using terminal services

Once Terminal Services is correctly configured, you can start using it. To remotely access the

eXlerate Terminal Services system, start the 'Remote Desktop Client'.



Figure 71 Remote Desktop Connection

Enter the remote computer name or IP address and click 'Connect' to establish the connection with the eXlerate Terminal Services system.

By default, the size of the desktop will be adjusted to the screen resolution of the client computer. The eXlerate application may however have been developed for a specific resolution. To open a remote desktop connection with a specific screen resolution you can use the following command-line:

```
mstsc /v:<host> /w:<width> /h:<height>
```

Example:

```
mstsc /v:10.0.0.105 /w:1920 /h:1080
```

16. Multi-lingual systems

Multi-lingual systems can change the display language during runtime. In this chapter you will learn how to configure multi-lingual systems.

16.1. Windows language packages

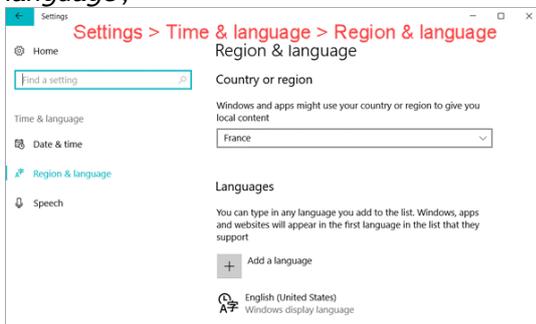
Dependent on the installed version of Windows, additional languages need to be installed. It is not necessary to have a localized version of Windows installed; any version of Windows will do. The localized versions of Windows translate OS texts such as: Start Menu, Control Panel, Print dialogs, etc. into a specific language.



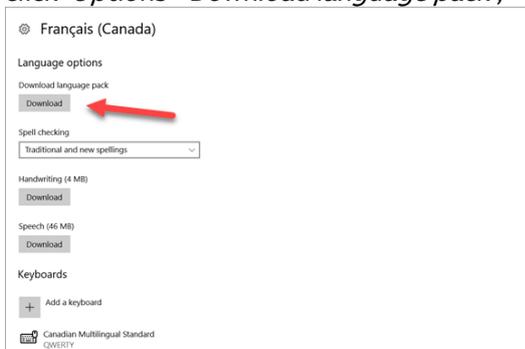
Install the Windows language packages to have the required code-page containing the characters for all the languages.

The installation depends on the used Windows version and build. For Windows 10, perform the following steps:

- Go to 'Settings - Time & Language - Region & language';



- Select a region, then click 'Add a language';
- Choose the language you need.
- Click the language pack you just added, then click 'Options - Download language pack';



- Restart the computer when a new language is installed.

16.2. Multi-lingual application

eXlerate supports multiple languages in an application. It requires the worksheet called 'xLanguage'. The layout of this worksheet is also fixed and can be extended with additional languages and custom texts.

Key	English	Nederlands	Deutsch	中文
Flag image name:	uk.gif	Netherlands.gif	Germany.gif	China.gif
System				
Alarms\Column\BlockCount	BlockCount	BlockCount	BlockCount	总分块数
Alarms\Column\Deadband	Deadband	Dodeband	Deadband	死区
Alarms\Column\Delay	Delay	Vertraging	Verzögern	延时
Alarms\Column\Description	Description	Omschrijving	Beschreibung	说明
Alarms\Column\Format	Format	Formaat	Format	格式
Alarms\Column\ID	ID	ID	ID	识别号
Alarms\Column\LastValue	Last Value	Laatste waarde	Letzter Wert	终值
Alarms\Column\Limit	Limit	Begrenzing	Grenze	范围
Alarms\Column\Location	Location	Plaats	Ort	地点
Alarms\Column\Name	Name	Naam	Name	名称
Alarms\Column\Priority	Priority	Prioriteit	Priorität	优先
Alarms\Column\State	State	Staat	Zustand	状态
Alarms\Column\Timestamp	Timestamp	Tijdstip	Zeitstempel	时间标记

Figure 72 Languages sheet

There are two methods to add multi-lingual support to an application.



Copy an 'xLanguage' worksheet from an existing application containing the translations if available.



If not available, run the **Language wizard** to generate a new 'xLanguage' worksheet.

The newly created worksheet will have no default formatting, so you need to apply it manually.

16.2.1. Language sheet

The 'xLanguage' sheet contains the text to display for the different languages. A language with its text is grouped into columns: each language consists of its own column. The first three columns of the 'xLanguage' sheet are fixed.

1. Class used for grouping sections
2. Key Internal key used for configuring multi-lingual text. More details below.
3. Default Default text (English)

Additional columns that have a non-empty value in the 1st row, are considered additional languages.

Additional rows contain the internal used 'language keys' with the corresponding text to display for each language. When text is not filled in, the text of the "Default" column will be used.

Language keys usually consist of several words separated by the backslash characters '\', like "General\Dialogs\SaveButton". This syntax is not

obligatory, but it is recommended as it improves readability and extensibility.

Some language texts support dynamic keywords which are replaced by specific values when the text is displayed. For instance, the language key “Alarms\Messages\AckGroup” supports the keyword ‘%USER%’, which when displayed is replaced by the actual name of the logged in user.



To add languages, create a column in the language sheet and set the language name. Then add the translations into the rows.

The language worksheet contains the ‘System’ class. These texts are used for dialogs, log texts, notifications, etc. which are generated by eXlerate rather than the application. All the keys in the ‘System’ class are fixed and cannot be changed.



You can extend the language sheet with user defined texts, to be used on display sheets and VBA user forms.



Start with a new class for your text keys to prevent these are deleted as the ‘system’ class is updated by the ‘Language wizard’.

16.2.2. Multi-lingual tag database

The Tag database supports multi-lingual tag and alarm descriptions. Additional language texts can be added to the Tag Database directly and don't need to be configured through the ‘xLanguage’ sheet.



To add multi-lingual tag descriptions, insert a column into the Tag database and give it the header “*Description_<Language>*”.

The “*<Language>*” section should be replaced by the name of the language (e.g. “Dutch”, “Russian”, “Chinese”). If the currently selected language is “Dutch”, the tag descriptions from the column “*Description_Dutch*” will be used for event logging and displaying. The column “*Description*” is used if the selected language is not explicitly configured or “Default” is selected.

The same mechanism applies to alarm descriptions. If alarm descriptions are explicitly configured using the “*AlarmDesc*” column, the additional languages can be added by adding new columns and appending the “*_<Language>*” (e.g. “*AlarmDesc_Dutch*”).

16.2.3. Language selection

The named cells *xLangIndex*’ and *xLangSelection*’, located on the internal ‘xWizard’ sheet, contain the currently selected language. The first is the index number, the 2nd is the language name in text, e.g. “English”. You can use this name to read the current language and to set the language.



To select a language, write the language name ‘*xLangSelection*’

Within VBA:

```
exRange("xLangSelection").Value = "Dutch"
```

From a worksheet:

```
=exCopy("Dutch", xLangSelection)
```

16.2.4. Multi-lingual displays and forms

The function ‘*exLanguageText()*’ returns the text associated with a Language Key for the currently selected language. If the text is not available for the selected language, the default text is used.

On worksheets, the function uses the trigger ‘*xLangRecalc*’ which triggers a recalculation every time a different language is selected. This trigger is not required when using the function in VBA.



To extend display sheets and VBA user forms with multi-lingual support, use the ‘*exLanguageText()*’ function.

```
=exLanguageText("MyText", xLangRecalc)
```

16.2.5. Multi-lingual buttons

For multi-lingual text on display buttons, the alternative mechanism can be used. Instead of using the ‘*exLanguageText()*’ function, the Button table uses the CHOOSE(...) function with ‘*xLangIndex*’ to select the button text from multiple columns containing language texts.



Add the button table language columns in the same order as the language columns in the ‘*xLanguage*’ worksheet.

If this is not the case, the indexes will be different, and the wrong language text will be selected.



When a new language is selected, call VBA function ‘*exSetButtonText()*’ to update the texts on all the buttons in the Button Table.

```
Sub SelectLanguage ()
    exRange("xLangSelection").Value = "Dutch"
    exSetButtonText
End Sub
```

17. Daylight saving time

Systems that require to support Daylight Saving Time (DST), require special attention to avoid fiscal integrity problems with respect to these changeovers.

Instead of using the Windows clock adjustment, let the eXlerate application adjust the computer(s) system time during DST change-over. eXlerate adjusts the time in two half hour steps, avoiding full hour changes that may invalidate hourly based counters and averages.

DST change-over in eXlerate

When the DST officially starts, the clock is set forward for one hour, e.g. at 02:00 at night the time is adjusted to 03:00.

In eXlerate, this changeover takes place in two half our steps:

- At 02:15, the clock is adjusted to 02:45
- At 03:00, an hourly transition takes place.
The data at 03:00 contain data for 30 minutes:
02:00-02:15 + 02:45-03:00
- At 03:15, the clock is adjusted to 03:45
- At 04:00, an hourly transition takes place.
The data at 04:00 contains data of the 2nd half hour: 03:00-03:15 + 03:45-04:00

The net result is that the clock is adjusted a full hour and that the fiscal reports contain two hours each with data of a half hour.

When the DST officially ends the clock is set backward for one hour, e.g. at 03:00 at night the time is adjusted to 02:00.

In eXlerate, this changeover takes place in two half our steps:

- At 02:45, the clock is adjusted to 02:15
- The 2nd time the clock is 02:45, the clock is adjusted to 02:15 again.
- At 03:00, a normal hourly transition takes place.
The data at 03:00 contains the data for 2 hours
(02:00-02:45 + 02:15-02:45 + 02:15-03:00)

The net result is that the clock is adjusted a full hour and that the fiscal reports contain one-hour record with 2 hours of data.

This behavior is generally accepted by the industry.

Configuration

 Turn off the Windows *'adjust for daylight saving time automatically'* when DST changeovers are configured in eXlerate.

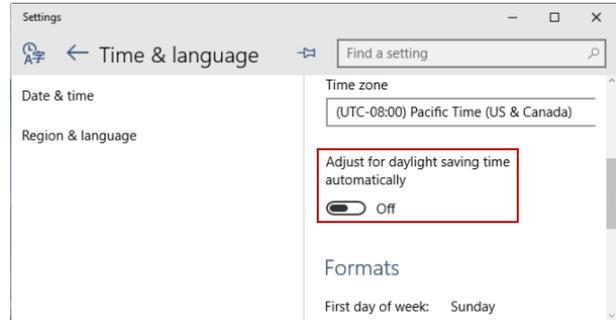


Figure 73 Windows DST setting

In your application, the settings for DST changeovers are defined in the *'rTimeTable'* on the sheet *'xTables'*.

TimeTable						
DST Settings						
	Month	Week	Day	Hour	Intermediate	Change DST
xDST_Start	3	5	1	2	31/Mar/2020	29 Mar '20 02:00
xDST_End	11	5	1	3	30/Nov/2020	29 Nov '20 03:00
rTimeTable						
Smon	Sday	Shr	Wmo	Wday	Whr	Reconfigure
3	29	2	11	29	3	4

Figure 74 Time table

This table contains 2 sections:

- DST settings
The first part contains the date (month, week and day of the week) and hour when DST starts and DST ends. These are used to calculate the actual date and time of DST changeovers.
- Named range *'rTimeTable'*
This range contains the values that are internally used by eXlerate and are derived from the DST settings.

 The application *'DaylightSaving.xlsx'* contains an example on the daylight-saving time configuration for in eXlerate.

In the example application, the DST settings are named cells.

 Use the named cells in your application to allow a user to change DST settings in run-time.

- xDST.Start.Month
- xDST.Start.Week
- xDST.Start.Weekday

- xDST.Start.Hour
- xDST.End.Month
- xDST.End.Week
- xDST.End.Weekday
- xDST.End.Hour

When any of these values changes, either the year changes or due a user entered new values, the DST change-over is reconfigured by the function *'exReconfigureSummerWinter()'* on the left side of the *'rTimeTable'* range.



Set in VBA function *'OnBeforeActiveMode()'* to call *'exEnableSummerWinter()'* to initialize the settings for DST changeovers.

18. Document revisions

Revision A February 2012

- Initial release for eXlerate 2010
- Added 'Import Sheets' and 'Advanced Replace' tools.

Revision B December 2016

- Update to eXlerate 2016.
- Update to ABB lay-out.
- New document code: COI/eXL2016-EN.

Revision C September 2018

- New document code: CM/eXL-EN.
- Reintroduce revisions chapter.
- Provisional support for MS Excel 2019 added.
- Windows 8 removed from software requirements.
- Clarified behavior of limit alarms at set points.

Revision D April 2019

- Support for MS Excel 2019 added.
- Updated chapter Relational Databases.

Revision E April 2020

- List of used network ports added.
- Updated contact address and added reference to installation manual.

Revision F January 2021

- Combining and rewriting the two reference manuals into a single configuration manual.
- Split installation and configuration sections in separate documents.

Revision G August 2022

- Added additional safety precautions for VBA code.

Appendix A. Troubleshooting

There are circumstances in which you need to troubleshoot an application. This could, for example, be due to issues with communication or issues with the computer hardware. There are several things you could do in these cases.

A.1. Do's and don'ts

Following this summary of the best engineering practices will, in many cases, improve stability and performance of your eXLerate application. More detailed explanations are shown in the previous chapters of this manual.

- Start with latest template application;
- Limit the amount of named references;
- Check in Name Manager the scopes and errors;
- Check for circular references and errors;
- Don't use volatile functions;
- Don't use 3D effects on shapes;
- Don't use references on shapes;
- Limit merging cells as much as possible;
- Don't use ActiveX combo boxes;
- Analyze your database queries;
- Clean up objects in VBA;
- Use eXLerate functions, like exMsgBox and exRange, replacing excel VBA functions;
- Use modeless user forms, especially when called from the event loop;
- Check that UDF's are not continuously called from worksheets;
- Limit the calling functions from event loop, especially for second-based events;
- Check performances of VBA functions;
- Use asynchronous database queries when possible;

A.2. Communications

See section *'xlConnect'* for details on debugging and logging communications and the explanation of the communication messages of the different protocols.

A.3. Reports

One of the most important tasks of the system is to generate the fiscal reports. When the system is down or eXLerate/your application has issues, you can still access the stored reports when you have sufficient privileges. All reports are stored in the folder as specified in the application's shortcut properties in the control center.

When reports are not generated as expected, perform the following checks:

- Check the log file messages
- The report template file is available in Xlrx\Cache folder
- If the report template file is missing, check the log file for messages when the application is started.
- Check the application *'rReportTable'* that all entries are valid:
 - Report names are defined
 - Worksheet names are defined
 - Worksheets are available in the application
 - Period to generate the report exists
 - File name doesn't contain invalid characters:
 - < (less than)
 - > (greater than)
 - : (colon)
 - " (double quote)
 - / (forward slash)
 - | (vertical bar or pipe)
 - ? (question mark)
 - * (asterisk)
 - For subdirectories use "\" (backslash)
 - Sheet name is limited to 31 characters
 - Sheet name doesn't contain invalid characters
 - [(square bracket)
 -] (square bracket)
 - < (less than)
 - > (greater than)
 - : (colon)
 - " (double quote)
 - / (forward slash)
 - \ (backslash)
 - | (vertical bar or pipe)
 - ? (question mark)
 - * (asterisk)

A.4. Windows event viewer

The Windows event viewer logs may indicate issues with Windows, applications, and hardware (drivers).

To check the Windows event viewer, use the following steps:

- Press <⊞-R> to start the “Run” prompt
- Type ‘eventvwr’ (without the quotes) and press <Enter>

A.5. Diagnostic information

Diagnostic tools are available when you have installed eXlerate. You will find these in the Windows start menu – All Programs – eXlerate – Tools. When you run the diagnostic tools, it will assemble diagnostic data from the computer:

- A snapshot of the environment variables
- A copy of Microsoft System Information (if the program can be found on the system) including hardware information, driver software information, Office versions etc.
- A registry extract from the eXlerate sections
- A full directory listing of drives C - F.
- Event files from the event viewer. These files sometimes show network problems, and security message that may have an impact on our software.
- Log files from the last 30 days
- Crash dump files, should they exist

Once completed, the user is invited to copy any additional files to the c:\tmp\exlerate directory, using Windows explorer. These files will be included in the CAB files. These files can be mailed to our tech support for further analysis.

A.6. Performance monitor

You can use Windows Performance Monitor to analyze data, such as processor, hard drive, memory, and network usage. To open Performance Monitor:

- Press <⊞-R> to start the “Run” prompt
- Type ‘perfmon’ (without the quotes) and press <Enter>

You can add counters to monitor virtually anything on your computer. Once you have configured all

the counters you want to monitor, you can also customize various aspects of the data shown in the graph

Since eXlerate version 4.2.5, the eXlerate performance is periodically logged (once per hour).

You can also add performance indicators into your application, using the worksheet function ‘exGetPerformanceInfo()’. The function returns

- Average CPU usage since previous call
- Memory usage
- Handle count

You can set trending and alarming on these values by using this function in the ‘xTagDB’.

Appendix B. Constants

Data types

xBit	1
xByte	2
xShort	3
xWord	4
xUInt24	5
xLong	6
xDWord	7
xChar	8
xRevDWord	9
xFloat	16
xRevFloat	17
xDouble	18
xShortFloat	19
xIntelFloat	20
xWordFloat	21
xRevDouble	22
xBCD	32
xTimeDate	33
xTimeStamp	34
xAdcFloat	37
x10kFloat	38
xBitInQWord	39
xLowQWord	40
xString6	64
xString12	65
xString24	66
xString10	67
xString80	68
xString	69
xString8	70
xString16	71
xVariant	80
xVarArray	81

Query table options

xBlockWrites	1
xNewDataOnly	2
xTransparentRead	4
xForcedWrites	8
xNoReadOnce	16
xItemUpdates	32
xNoSleepAll	64
xWriteOnly	128
xWriteAll	512

Update constants

xUpdateNever	1
xUpdateAlways	2
xUpdateConditionally	3

Periods

xSec	1
xMin	2
xHour	3
xDay	4
xWeek	5
xMonth	6
xQuarter	7
xYear	8

Pre-defined colors

xBlack	0
xWhite	1
xRed	2
xGreen	3
xBlue	4
xYellow	5
xMagenta	6
xViolet	7

Editing types

xWholeNumber	1
xDecimal	2
xText	3
xList	4
xDate	5
xTime	6

Editing limits

xMinimum	1
xMaximum	2

Editing targets

xTargetNone	1
xTargetCell	2
xTargetName	3
xTargetComm	4
xTargetAlarmLimit	5
xTargetAlarmDeadband	6
xTargetAlarmDelay	7

General

xAddress	1
xFormat	2
xRawFormat	3
xFormula	4
xValue	5

Appendix C. License model

Licenses for eXLerate are available depending on the following factors. Please contact ABB BV for details.

- eXLerate operating mode:
 - Development
To develop applications and test and run communications for up to one hour.
 - Runtime
For runtime use of applications, with communication, live data, visualization, etc.
 - Development + Runtime
To run and maintain applications.
- Communication Protocols:
 - Flow-X Client
 - Modbus Client (TCP)
 - Modbus Server (TCP)
 - Modbus Master (serial)
 - Modbus Slave (serial)
 - OPC Client
 - OPC Server
 - HART Master
 - HART Slave
 - SLIP+
 - Uniform
- Number of I/O tags in your application:
Each value communicated with an external device is a tag, see section '*Tag Database*'.
 - 75 tags
 - 150 tags
 - 300 tags
 - 750 tags
 - 1500 tags
 - 3000 tags
 - 6000 tags
 - 32765 tags
 - 1000000 tags
- Network arrangement:
 - Stand-alone
 - Server
 - Client
 - Terminal services
- Additional functions:
 - Foreign Database
 - Flow-Xpert Libraries
 - Virtual Flow Computer
 - Flow-Xprint Virtual printer

ABB B.V.
Measurement & Analytics
Achtseweg Zuid 151A =
5651 GW Eindhoven
The Netherlands
Phone: +31 40 236 9445
Mail: nl-spiritit-sales@abb.com

ABB Malaysia Sdn Bhd.
Measurement & Analytics
Lot 608, Jalan SS 13/1K
47500 Subang Jaya
Selangor Darul Ehsan, Malaysia
Phone: +60 3 5628 4888

abb.com/midstream

ABB Inc.
Measurement & Analytics
7051 Industrial Boulevard
Bartlesville OK 74006
United States of America
Phone: +1 800 442 3097

ABB Limited
Measurement & Analytics
Oldends Lane, Stonehouse
Gloucestershire, GL10 3TA
United Kingdom
Phone: +44 7730 019 180



We reserve the right to make technical changes or modify the contents of this document without prior notice. With regard to purchase orders, the agreed particulars shall prevail. ABB AG does not accept any responsibility whatsoever for potential errors or possible lack of information in this document.

We reserve all rights in this document and in the subject matter and illustrations contained therein. Any reproduction, disclosure to third parties or utilization of its contents – in whole or in parts – is forbidden without prior written consent of ABB.

© ABB 2020