

## FOR DEVICE DEVELOPERS - not intended for end-users

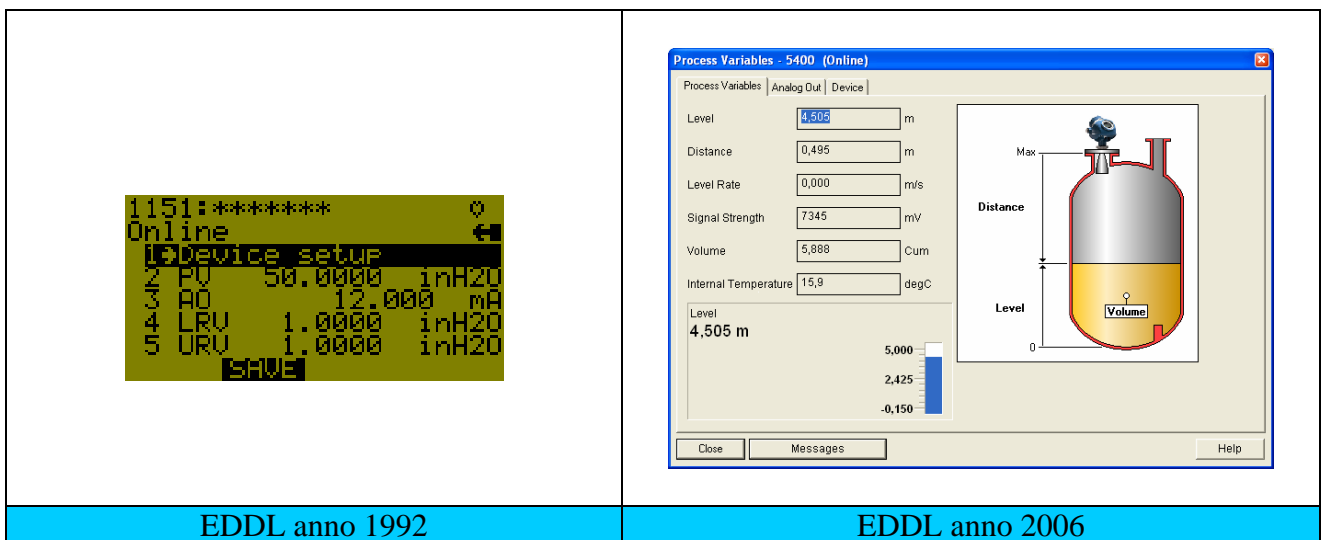
### Improving User Experience with EDDL

EDDL often gets accused of not permitting device developers to decide how device management software displays their device to the technician. This paper sets the record straight and highlights new capabilities for device information discovery and retrieval. EDDL is an integral part of existing fieldbus protocols and integrates with devices and systems providing powerful new capabilities for using information from networked devices, a portal into a broad universe of device information. EDDL is fully capable of dealing with device data using its intelligent graphical user interface and hierarchical organization, providing device manufacturers with extensive support options for device page customization.

Since 1 January 2009 all Fieldbus Foundation check-marked control system and field communicators must support EDDL enhancements to pass the new Host Registration Program, including graphical enhancements. All product managers should pay attention because it means leading control systems will support EDDL for wired HART, WirelessHART, and PROFIBUS devices as well since it is the same standard. All leading system suppliers have either already passed the Fieldbus Foundation test or demonstrated support for EDDL enhancements. Thus it makes sense for device manufacturers to provide enhanced EDDL files for all their devices now. Device developer can reuse their traditional DD files, simply adding graphics. There is no need to wait for the next generation of device hardware or software, upgrade from DD to enhanced EDDL can be done for any HART, FOUNDATION fieldbus, or PROFIBUS device.

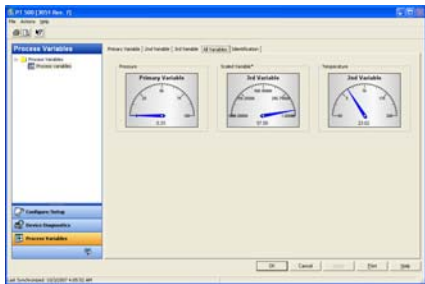
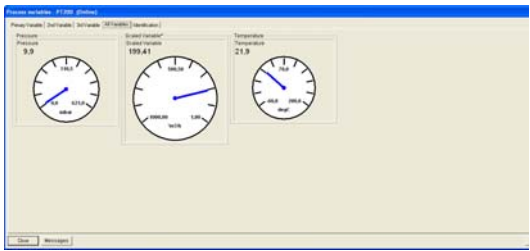


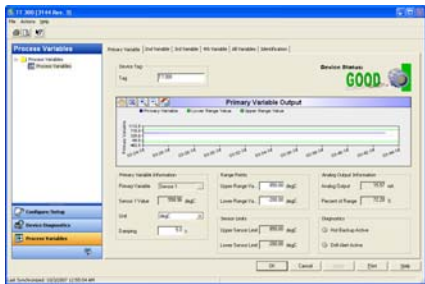
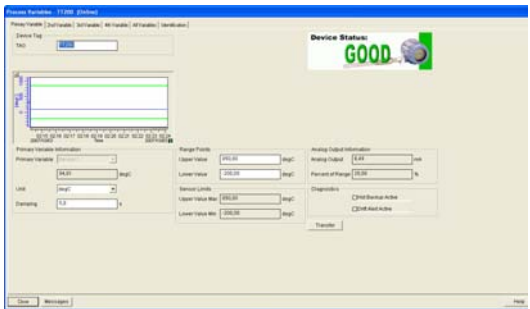
### Background

The HART protocol enabled a single handheld terminal to connect to instruments of all kinds from different manufacturers. However, it was EDDL technology introduced in 1992 that allowed the information in these devices to be transported and displayed on screen. EDDL became the standard that was so simple and compelling that any and every device developer could innovate on top of it. It quickly scaled around the world to device developers from Sao Paulo to Tokyo. It was also adopted as part of FOUNDATION fieldbus, PROFIBUS, and later OPC. EDDL enables a single software to be used for all devices, using multiple protocols, for commissioning, setup, calibration, diagnostics, and other maintenance tasks.



In 2004, EDDL became an international standard: IEC 61804-2. At the same time it was agreed that graphics capabilities had to be standardized as different system vendors began creating different proprietary solutions, thus hampering interoperability and increasing development and support cost. As a product manager you may have experienced cases where your device only displayed as a long disorganized list of parameters, not as nice as some other devices. Device developers were asked to develop software for the device to display nicely, for every different software application your customers use, for every device you sell, and then again when improvements were made. To eliminate the need for proprietary graphics solutions, graphical capabilities were added to EDDL and released as an international standard in 2006: IEC 61804-3.

**Table 1 The device display content and general organization is the same in different systems**

System A	System B
	
	
	









The IEC standard provides easy to grasp graphical metaphors like gauges and recorders. Using the graphical capabilities of EDDL your device can be displayed with as much pizzazz as sophisticated devices such as radar level transmitters, machinery health monitors, and variable speed drives. Completely transparent for data location and retrieval process, EDDL does everything it can do to avoid burdening the user with technical details of the data communication, to allow the user to focus all attention on the task at hand - setup, calibration, or diagnostics.

Screen captures in this document are taken using EDDL software from Emerson Process Management, HART Communication Foundation, National Instruments, and Siemens.

## Integrate on any host

In order for a device to be able to integrate with any control system or handheld communicator, with its full range of capabilities, device suppliers must provide an Electronic Device Description (EDD) for the devices. EDDs can be developed for devices currently available for sale, as well as devices that have been retired, as a way to extend their useful life.

**Table 2 EDDL is already an integral part of HART, FOUNDATION fieldbus, PROFIBUS protocols**

Protocol		EDDL Specification / Profile		
HART	 	HCF_SPEC-500 and HCF_SPEC-501		
FOUNDATION fieldbus		FF-900 and FF-901		
PROFIBUS		Guideline 2.152 volume 2 and 2.362		

EDDL is a standard that enables true innovation. The EDDL standard allows the device developer to focus on where the real value lies: making the device easier to use. Using EDDL there is no need for device developers to modify, compile, and validate software every time a new version of Windows or service pack is released.

Before the graphical capabilities were added to EDDL, use cases involving all kinds of process control instruments in the different phases of their life cycle were collected. The functionality provided by proprietary software drivers was studied. With this knowledge in hand, graphical capabilities were added to EDDL.

Version 5.1 of EDDL includes "device-level access" (a.k.a. "cross-block") functionality that permits display of parameters from multiple blocks on the same screen. This will further simplify use FOUNDATION fieldbus devices. The technician no longer has to worry about a parameter in a FOUNDATION fieldbus device being located in the resource block or one of the transducer blocks, just focus on the task at hand "no block": calibration, diagnostics, parameterization etc.

The original EDDL technology from 1992 works with all hosts and will continue to work. However, by adding graphical capabilities to EDDL, device developers can easily increase customer value by making the device easier to use and enable new functionality. The original EDDL technology was not wiped-out. The language was simply extended. Therefore, all existing EDDs still work with today's control systems and will continue to work on into the future. It also means that adding graphics to existing EDDs is easy. The new graphics capabilities have been added while retaining all the much appreciated advantages of EDDL: robust, externally accessible

information, single universal solution, consistent display of devices, investment protection, no version conflicts, cross-platform compatibility, easy integration and removal, no licensing, and certification.

### System files vs. device files

Standard EDDL graphical enhancements eliminates or reduces the reliance on system specific files because system specific graphics is no longer required. This makes systems interoperable with more devices, new devices can be adopted by system owners sooner. Device manufacturers are in control on how their device is displayed in system, and better in control of how new versions can be integrated to systems. System specific files may be required for the alarm management system which is a system feature, not a device feature.

### Usability: Make your device more user friendly

By providing EDDs, device developers can make device setup easier and more intuitive. For example, device developers can display level as a vertical bar-graph to technicians. Device developers can include images that illustrate high and low alarms, demonstrate hysteresis, and provide an image of the product itself. With EDDL it is also possible for device developers to organize the parameters in the device with the most common parameters "in front" and the advanced parameters on a tab "in the back" and grouping of parameters according to functionality such as measurement, alarms, and diagnostics. As customers start discovering the advantages of EDDL they will start demanding "EDDL compliant to IEC 61804-3" for their devices. Device developers can use EDDL to create a rich and easy to use interface for their devices.

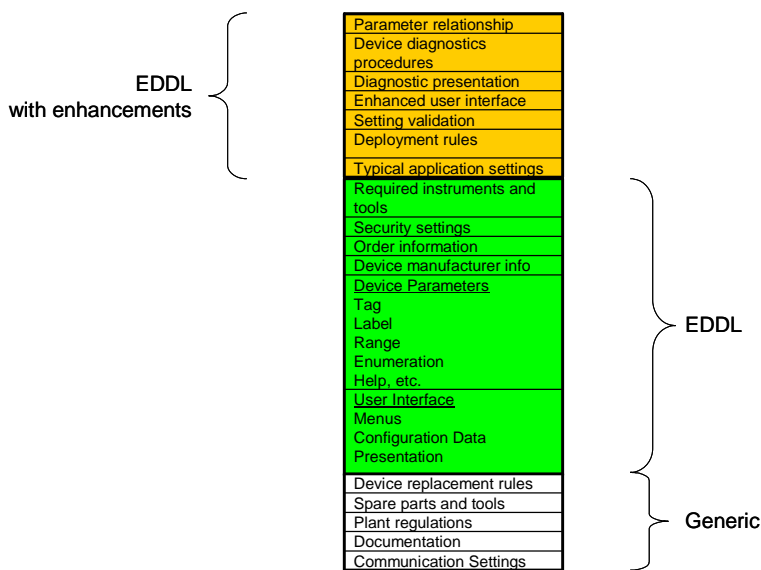


Figure 1 More functionality added with EDDL enhancements

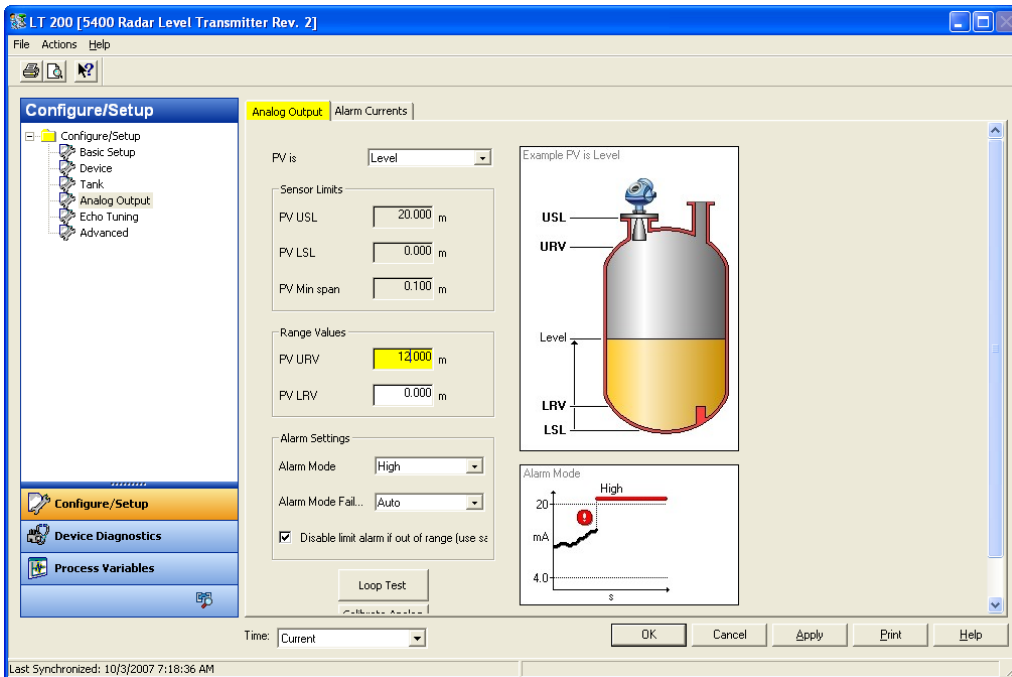
Keep in mind that many generic device management functions do not need EDDL support. And it does not require communication with device i.e. requires no decoding or display of data. For example, the manual or other documents for a device open as a link to a file on the system hard disk by clicking on the device icon, they are not stored in the device, and they are displayed using readers such as Adobe Acrobat.

The graphical elements are generic and very flexible, matching the capabilities of new devices that will no doubt be appearing in the industry the next several years. Device developers need not be concerned with printing configuration, charts, or graphs as this functionality is provided by the device management software. Using EDDL the device management software can decode all data in the device, and from there it can print it in simple table format or structured according to the menu system for the device. Similarly, the device developer need not be concerned with saving configurations or maintaining an audit trail since this is taken care of by the device management

software. Same goes for comparing/reconciling the database with the actual device. Likewise, the device developer need not be concerned with Excel export. Using EDDL the device management software can decode all data in the device, and from there it can export it to Excel in simple table format or structured according to the menu system for the device.

**VARIABLE: Defining the data**

The device developer need not be concerned with indicating which value can be edited or is read-only, or indicating which value has been changed and need to be downloaded. This is handled automatically by the device management software, for example "grayed out" for read-only and yellow for changed values.

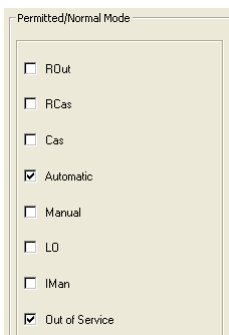


**Figure 2** read-only parameters as well as changed values are differentiated from others

The device developer need not be concerned with inconsistent data when multiple users work on different machines accessing same information. The device management software takes care of it. Similarly sending values, and discarding changes on cancel, is also taken care of automatically. Lastly, device developer need not be concerned with online or offline operation or making sure user is logged out after a period of inactivity.

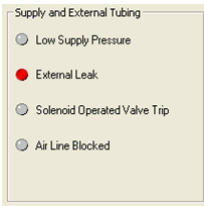
**Bit Enumerated**

By using bit enumerated variables the device developer will get the variable rendered by device management software as a checkbox with associated description on the display. The device developer is able to mask the bits in a variable to only show those bits which are applicable in a particular menu. If the bits can be written, they are automatically rendered as check boxes.



**Figure 3** writable bit-enumerated parameters rendered as check boxes

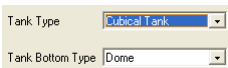
If the bits are read-only, their status is automatically displayed as "indicator lights" that are either on or off.



**Figure 4** read-only bit-enumerated parameters rendered as indicator lights

### Enumerated

By using enumerated variables the device developer will get the variable rendered by device management software as drop-down lists with associated text on the display.



**Figure 5** Enumerated parameters rendered as drop-down lists

### ***MENU: Organized display of information and access to functions***

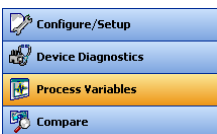
Menus have existed in original EDDL since 1992 but have been drastically enhanced in IEC 61804-3. The menu construct allows the device developer to logically organize the display of data from the device to make the device easy to use and to make it easy to navigate the information as well as to invoke functions such as calibration trim etc. For instance, put "advanced" functions or diagnostics in different tab from "basic" or in a pop-up window in order to make sure the basic page is not cluttered with rarely used functions only understood by experts.

There are six menu styles that device developers can make use of to organize data:

- Dialog (Box)
- Window (Form)
- Page (Tabs)
- Group (Frame)
- Menu (Pop-Up, Pull-Down)

It is really only the "Menu" style of menu that appears as an actual Windows menu. The other styles really appear as Windows forms, dialog box, tabs, frames, and all the other familiar Windows controls used to enhance the look & feel for the user. The device developer can organize related parameters together on tabs, columns, and in frames. Parameters that should only be used in the factory may be hidden. The most important parameters may be shown first and the rarely used parameters may appear in "advanced" menus that do not clutter the display.

It is possible to define two different menu hierarchies; one that appears when displayed by device management software with large color monitors, and another that appears in handheld communicators with small monochrome screens. The IEC 61804-4 interoperability guideline guidance on the types of menu hierarchies that should be provided.



**Figure 6** Root menu using table style

Use the menu structure to organize data:

- By functionality: e.g. all diagnostics together
- By level of expertise: for expert or non-expert

- By frequency of use: common in front, rarely used a few clicks away

Hide/mask data irrelevant to the user, for example parameters only used in the manufacturing of the device.

**DIALOG Style**

Using the DIALOG style the device developer ensures the user must click a button to open a modal dialog box to reveal and edit parameters. In other words, the user must click OK before proceeding.

**WINDOW Style**

Using the WINDOW style the device developer ensures the user must click a button to open a modeless window to reveal and edit parameters. In other words, the user can switch back and forth from this window.

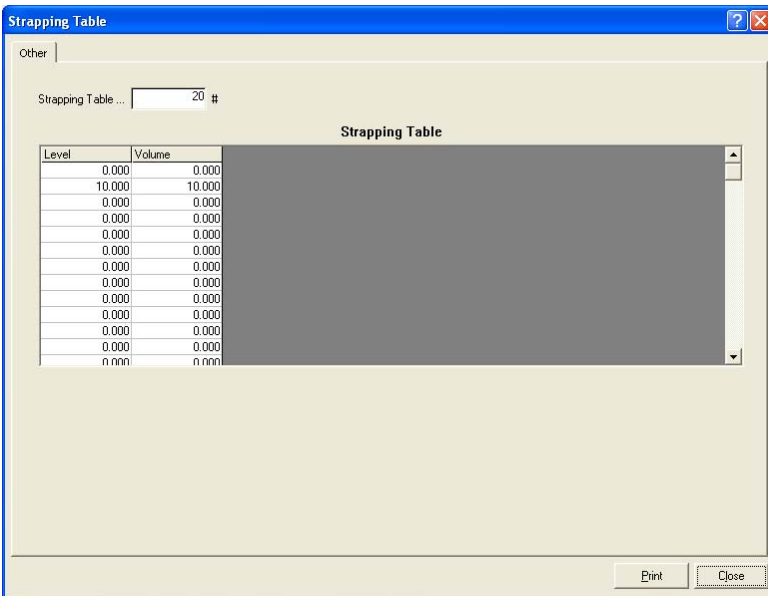


Figure 7 Window style menu

**PAGE Style**

Using the PAGE style the device developer ensures the user is presented with tabbed cards on which parameters and buttons are structured logically.

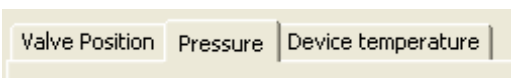


Figure 8 Page style menu renders as tabs

**GROUP Style**

Using the GROUP style the device developer ensures the parameters and buttons are logically organized within frames.

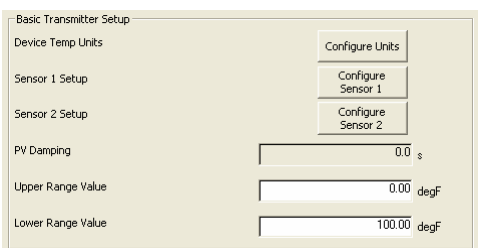
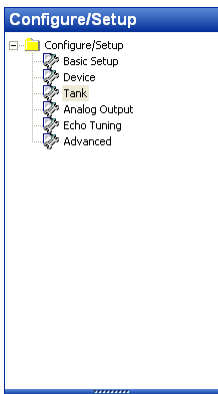


Figure 9 Group style menu renders as frame

## MENU Style

Using the MENU style the device developer ensures that the user is presented with a pull-down, pop-up menu, or navigation area and button tray.



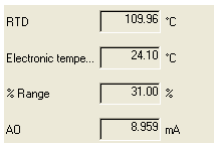
**Figure 10 Navigation area**

Using menus with conditionals makes it possible for the device developer to hide branches of the menu based on functions licensed or as applicable based on prior selections in the setup.

## Organizing variables and buttons

Through the use of menus, the device developer can organize the data on the screen and make the device easier to use. Device developers need not be concerned with configuration reconciliation (compare device to database) as this functionality is provided by the device management software.

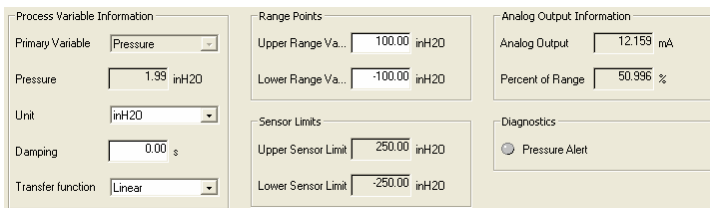
The device management software will ensure that parameter label, value, and unit are displayed side by side rather than in different places in a list.



**Figure 11 Label, value, and unit side by side**

The device developer can define an easy to understand LABEL for the variable that is displayed instead of cryptic parameter names, for example "Damping" instead of "PV\_FTIME".

The device developer can use COLUMNBREAK to organize the display side by side rather than in a single row as a very long display.



**Figure 12 Display as columns side by side**

The device developer can use SEPARATOR to introduce a dividing line, and ROWBREAK to start a new row.

Main Vibration		Auxiliary Vibration	
Motor Outboard Horizontal:	0.882058 Gs RMS	Motor Outboard Vertical:	-1.000000 Gs RMS
Motor Inboard Horizontal:	0.002552 Gs RMS	Motor Inboard Vertical:	-1.000000 Gs RMS
Motor Inboard Axial:	0.002572 Gs RMS		
Pump Inboard Axial:	0.002562 Gs RMS		
Pump Inboard Horizontal:	0.002558 Gs RMS	Pump Inboard Vertical:	-1.000000 Gs RMS
Pump Outboard Horizontal:	0.506970 Gs RMS	Pump Outboard Vertical:	-1.000000 Gs RMS
Other			
Motor Flux:	0.001500 v RMS	Tachometer:	0.000238 v RMS

**Figure 13 Break to a new row**

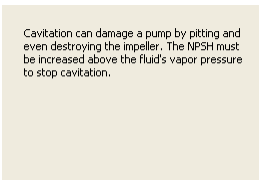
The device developer can use `VALIDITY` to hide a menu under certain conditions. For example, certain menus may only be valid if the device has been configured in a certain way, such as display settings are only valid when an LCD indicator is installed on the device or flow ranging is only valid when the transfer function of a differential pressure transmitter is set to square root.

Buttons need not be explicitly defined in EDDL. Whenever the device developer includes a method, edit display, dialog box, or window in a menu it automatically gets rendered as a button, which the technician can click to activate the method or open the edit display.



**Figure 14 Buttons need not be explicitly defined**

The device developer can include static text to explain a particular device function or to help understand the meaning of specific device diagnostics. For common messages and descriptions the device developer may make use of text from the standard dictionary to ensure consistency in user interaction. The text is available in multiple languages and displays automatically according to language selection. For non-standard text the device developer may include the same message in multiple languages.



**Figure 15 Device developer can include static text to guide technician**

### ***EDIT\_DISPLAY: Buttons opens dialogue box***

Edit displays have existed since 1992 but have been enhanced in IEC 61804-3. The device developer will likely use a `DIALOG` style menu instead because edit display is not as flexible, but EDDL still supports edit display to ensure backwards compatibility. Using an edit display the device developer can ensure the device management application displays parameters that should be displayed or edited together.

### ***CHART/SOURCE: Continuous trend and instantaneous illustration***

The device developer can chose to display a chart of multiple dynamic variables sampled periodically as a continuous trend against a time-axis on an on-going basis. For a technician this is often useful for detecting drift, intermittent problems, and other dynamics. Alternatively, the device developer can display the instantaneous value of a single parameter as a dynamic gauge or bar, which is useful when a finite range applies such as tank level, valve travel, or maximum pressure or temperature. Charts and sources allows the device developer to define the "paper" and "pens" respectively. The device developer can include multiple values in one chart. The device developer need not be concerned with zooming, scrolling, or printing as these functions are provided by the device management software.

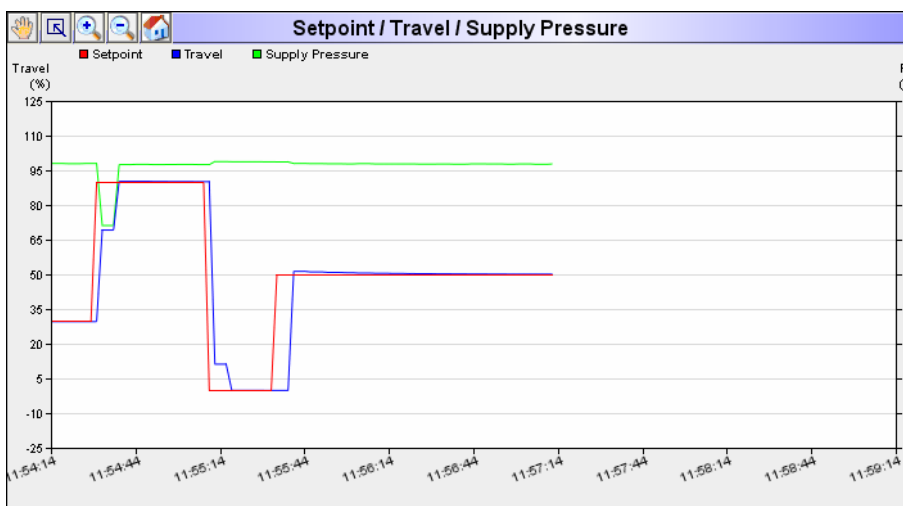
There are six types of charts that the device developer can chose from, within two sub-categories:

- Trend chart types showing change over time
  - Strip (scrolling)
  - Sweep (re-start)
  - Scope (erase)
- Instantaneous illustration types
  - Horizontal bar-graph
  - Vertical bar-graph
  - Dial gauge

### Strip Chart

The strip chart is a graphical metaphor of a recorder. Using the strip chart the device developer can display a scrolling chart. This allows the technician to perform trending on an ad-hoc basis to detect drift or intermittent problems. Strip charts may be included in process variable displays to show trend over periods of minutes to hours. This may be ideal for ambient conditions such as temperature, performance values such as dead-band and friction or torque. The device developer may choose to show multiple pens such as ambient temperature, desired valve position, actual valve position, and drive signal together on the same chart. The device management software will render charts with the same look for all devices regardless of protocol, manufacturer, or model, and will handle mouse clicks the same as well.

The device developer can use EDDL mathematical expressions with refresh to dynamically compute values from multiple inputs before display. Multiple values can be displayed in the same chart.

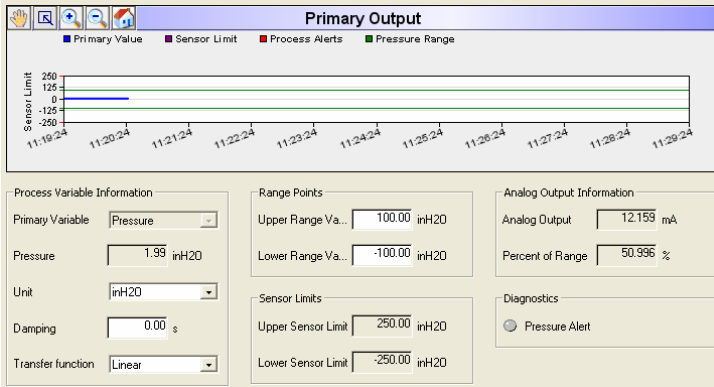


**Figure 16 Strip Chart with multiple pen sources**

The device developer need not be concerned about actually drawing the graph, panning, or zooming, because it is handled automatically by the device management software. Similarly, the device developer need not be concern with caching the source data in memory.

The device developer can set the desired sampling time for the chart using `CYCLE_TIME` and the time span using `LENGTH`. The device developer need not be concerned with the periodic sampling of the variables as this is handled automatically by the device management software.

Using `WIDTH` and `HEIGHT` the device developer defines the relative size of the chart on the display.

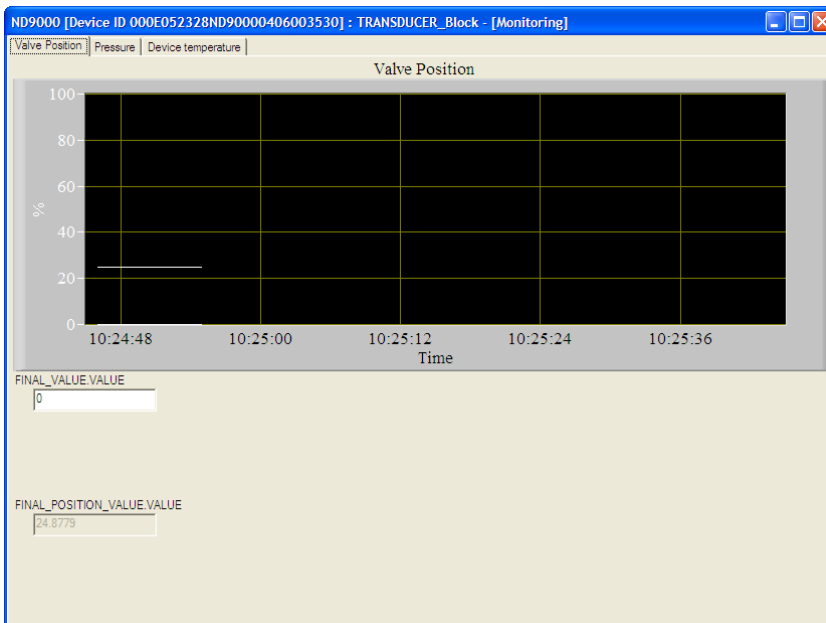


**Figure 17 Chart occupying with full width across several columns**

The device developer can plot multiple data lines "pens" in the same chart and can also include low-low, low, high, and high-high limit control lines. The device developer uses `LINE_TYPE` to distinguish different data lines and control lines. The device management software will render data lines and control lines with a consistent style (color, thickness, solid/dash, etc.) for all devices regardless of manufacturer or communications protocol.

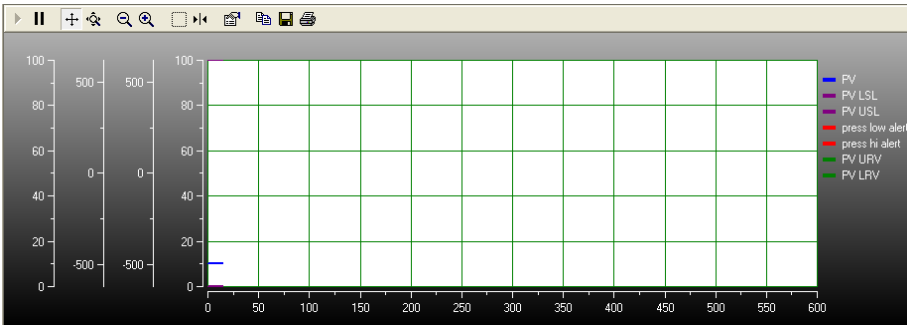
The device developer can, if necessary, force a specific line color, overriding the default pen style in the device management software, for each data line "pen" in the chart using `LINE_COLOR`. The device developer also has the ability to emphasize a particular data line, i.e., make one of the lines stand out because it's the most important one.

All the "pens" on the chart share a common time-axis but the device developer may assign different Y-axis (ordinate) for each "pen" if required, for example for primary, secondary, and tertiary values.



**Figure 18 Chart and axis are labeled**

For each Y-axis in a chart, the device developer can define minimum and maximum values (range), the engineering unit, and if the scaling shall be linear or logarithmic. The device developer defines the legend to be displayed for the "pen" using the `LABEL` of the source.



**Figure 19 Multiple Y-axis and legends for the pens**

The device developer need not be concerned about printing the graph, because it is handled automatically by the device management software.

The trend chart has no time limit but may not be suitable for long term trending. For trending over extended periods of time the technician should use dedicated trending applications. The device developer need not be concerned with making parameters available to external applications through OPC since the device management software takes care of that, including providing meta-data such as range and engineering unit.

Strip chart is most effective for values which are changing, less effective for values that change slowly or remain static.

### Scope Chart

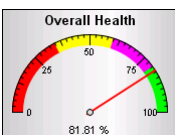
Using the scope chart the device developer can display a chart that is erased and restarted at the beginning whenever the plot reaches the end of the chart. Otherwise, it is identical to a strip chart.

### Sweep Chart

Using the sweep chart the device developer can display a chart that is overwritten from the beginning whenever the plot reaches the end of the chart. Otherwise, it is identical to a strip chart.

### Gauge Chart

The gauge chart is graphical metaphor of a mechanical gauge. Using a gauge chart the device developer can display a gauge-like display with scale and pointing needle indicating the instantaneous value of a dynamic variable, for example the measured value or the battery voltage in a wireless device. The device management software will render gauges with the same look for all devices regardless of protocol, manufacturer, or model. Using LINE\_TYPE the device developer can add different color bands to indicate too low, normal, ideal, and too high value. Gauges are useful to the technician who can step away from the handheld or laptop to adjust or apply on the device, yet see its alive from some distance.



**Figure 20 Rendering of gauge type instantaneous chart**

The device developer may make the technician's work easier by displaying multiple gauges for primary, secondary, and tertiary values on the process variable monitoring page. The device developer can use different sizes of gauges depending on importance of the variable. For example, primary variable gauge can be large while secondary and tertiary values can be smaller.

A gauge illustrates where you are between a low and a high reference point, and makes it easier to visualize if value is increasing or decreasing, and if it is fast or slow. The gage chart is most effective for values which are changing, less effective for values that change slowly or remain static.

**Table 3 there are small differences in gauge look between software from different manufacturers**

Emerson AMS	Emerson 375	Siemens PDM	HCF SDC625	National Instruments NI-FBUS
Device Management Software	Handheld Field Communicator	Device Management Software	Configuration & Diagnostics Software	Configuration & Diagnostics Software

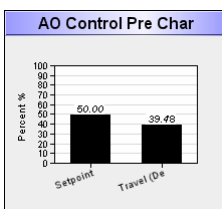
Devices appear the same within one device management application regardless of protocol, manufacturer, or type.

**Table 4 consistent appearance of gauge for devices with different protocols, manufacturer, type, and parameter**

Rosemount HART Pressure Transmitter	Siemens HART Temperature Transmitter	Rosemount Wireless Temperature Transmitter	Rosemount HART Temperature Transmitter	Fisher Fieldbus Valve Positioner
Pressure	Output Current	Battery Voltage	Temperature	Actuator Pressure

**Vertical Bar Chart**

Using the vertical bar chart the device developer can display a bar-graph with scale and a rising/falling column indicating the instantaneous value of a dynamic variable. For example, level, temperature, or wireless signal level.



**Figure 21 Rendering of vertical bar-graph type instantaneous chart**

The device developer can chose to display two bars in the same chart to enable comparison between desired value (setpoint) and actual value (process), or multiple bars side by side in the same chart to create a histogram.

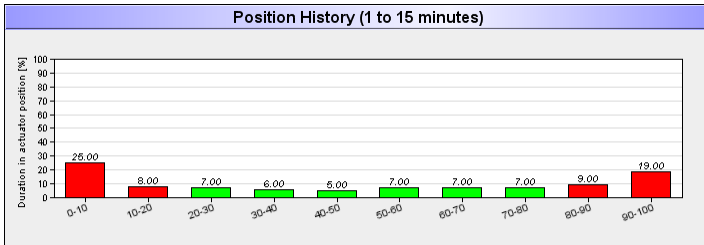


Figure 22 A histogram is a bar chart with multiple bars

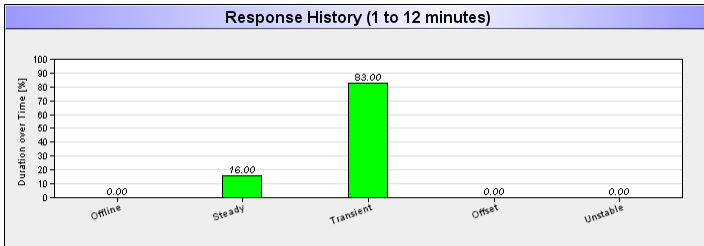


Figure 23 Bar chart with multiple bars

### Horizontal Bar Chart

Using the horizontal bar the device developer can display a bar-graph with scale and a sliding bar indicating the instantaneous value of a dynamic variable such as valve position. The device developer can chose to include multiple bars, e.g., to represent desired and actual valve position.

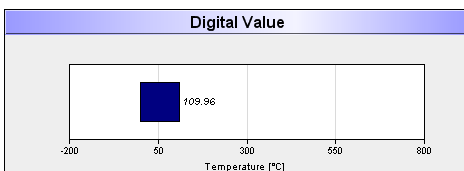


Figure 24 Rendering of horizontal bar-graph type instantaneous chart

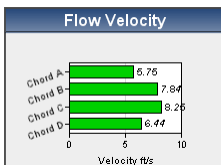


Figure 25 Rendering of multiple horizontal bar-graphs for comparison

### GRAPH/WAVEFORM: Plot curve from internal data

The device developer can chose to display a graph to visualize complex data sets. For a technician this is often useful for radar echo curve, valve signature, step-response, dynamic error band (hysteresis / positioner signature), drive signal, and output signal. Graphs and waveforms allow the device developer to define the "paper" and "pens" respectively. The device developer can plot multiple waveforms in one graph.

The device developer can enable the technician to retrieve a waveform stored in the device management software's persistent database as a reference to be plotted together with a curve just captured in the device. This would allow the technician to compare the reference waveform with the current waveform. This is an important way of detecting degradation in performance in certain devices such as valves. The device developer need not be concerned how the data is stored; it is all taken care of by the device management software.

The device developer can choose to display the data in one of four ways:

- XY plot
- YT plot
- Horizontal line
- Vertical line

Using WIDTH and HEIGHT the device developer defines the relative size of the graph on the display.

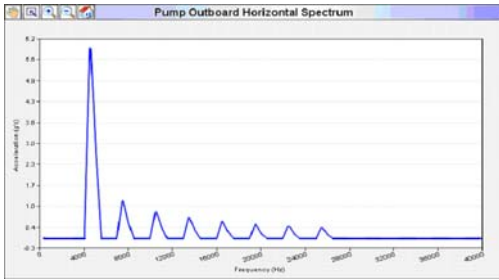
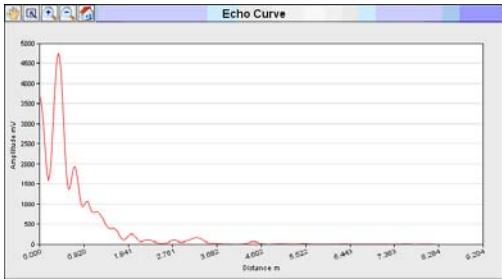
Optionally, the graph can be periodically updated. The device developer can define the refresh rate through CYCLE\_TIME. The device management will automatically refresh the graph, so the device developer need not be concerned with the mechanics of refreshing the data.

All the waveforms on the graph share a common X-axis (abscissa). For the shared X-axis in a chart the device developer can define minimum and maximum values (range), the engineering unit, and if the scaling shall be linear or logarithmic.

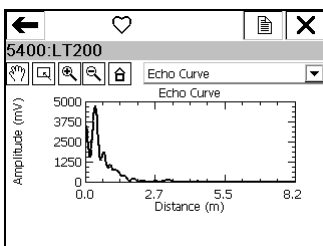
**XY Plot**

A scatter plot of multiple points on arbitrary X and Y coordinates can visually display a radar echo curve, valve signature, or motor vibration spectrum.

**Table 5 generic graphical elements are applicable to all kinds of devices for all kinds of functions regardless of protocol**

Device Type	Machine Health	Radar Level
Task	Diagnostics	Setup
Function	Vibration Spectrum	Echo Curve
		
X axis	Frequency	Distance
Y axis	Acceleration	Amplitude

The device developer need not be concerned about actually drawing the graph, because it is handled automatically by the device management software. Similarly, the device developer need not be concern with caching the waveform data in memory.



**Figure 26 Echo curve from radar level transmitter displayed in simple handheld communicator**

The device developer need not be concerned with zooming, scrolling, or printing as these functions are provided by the device management software.

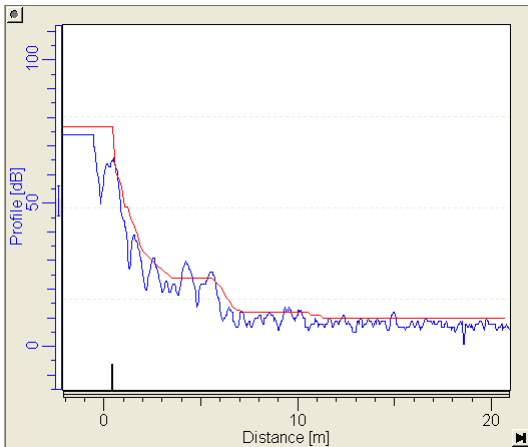
All the curves on the graph share a common X-axis but the device developer may assign different Y-axis for each waveform, if required. For each Y-axis in a graph the device developer can define minimum and maximum value (range), the engineering unit, and if the scaling shall be linear or logarithmic. The device developer defines the legend to be displayed by the device management software using the LABEL of the waveform.

The device developer can chose to plot multiple waveforms in the same graph and can also include low-low, low, high, and high-high limits control lines. The device developer uses LINE\_TYPE to

distinguish different data lines and control lines. The device management software will render data lines and control lines with a consistent style (color, thickness, solid/dash, etc.) for all devices regardless of manufacturer or communications protocol.

The device developer can, if necessary, force a specific line color, overriding the default pen style in the device management software, for each waveform in the graph using `LINE_COLOR`. The device developer also has the ability to emphasize a particular curve, i.e., make one of the lines stand out because it's the most important.

The device developer may also define a waveform as writable. That is, giving the technician the ability to adjust the waveform. For example, the device developer may give the technician the ability to edit a filter envelope for an echo profile curve to cancel false echoes, apply flow characteristics for a valve, or specify a tank linearization table.



**Figure 27 Graphical display and edit of echo profile in radar level transmitter**

The device developer need not be concerned with the mechanics of editing the values in the waveform, as it is handled by the device management software.

The device developer may also highlight multiple points of particular interest along the curve using the `KEY_POINTS`. This may include for example peaks or valleys. These key points may be highlighted via a crosshair, cross, spot, or some means.

#### **YT Plot**

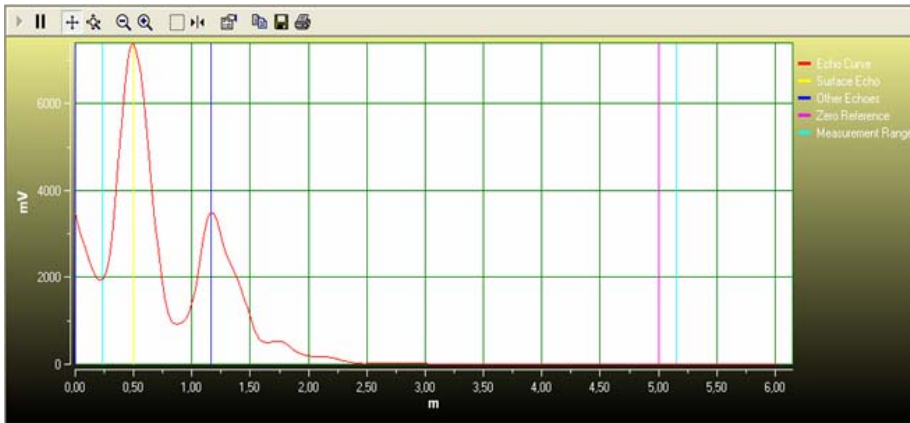
The device developer can make use of a plot of multiple points of arbitrary Y-coordinates but with regular time interval along the X-axis. The device developer can use it for example to visually display a capture of a fast step response of a valve. The device developer may also use this curve to display and edit ramping profiles for variable speed drives.

#### **Horizontal Lines**

The device developer can add one or more horizontal lines to the graph to indicate maximum or minimum limits along the Y-axis.

#### **Vertical Lines**

The device developer can add one or more vertical lines to the graph to indicate maximum or minimum limits along the X-axis. This may be used to indicate surface echo, false echoes, and range for a radar level transmitter.



**Figure 28** Curve with vertical lines

**FILE/LIST: Persistent data storage**

Using files the device developer can define a collection of data and enable the technician to store this data for future reference in the device management software's persistent database. This may include waveforms such as baseline valve signatures. The device developer can also enable the technician to retrieve the stored curve and compare against a waveform captured at a later date. This is an important way of detecting degradation in performance in certain devices such as valves.

A device developer need not be concerned with how the device management application stores the data. Stored data is associated with a particular device. However, the device developer need not be concerned with how the device management application uniquely identifies the device that is associated with the data.

**GRID: Data tables**

Some information is best shown and edited in tabular form like an Excel spreadsheet. A good example is a custom tank strapping table. The device management software will render tables with the same look for all devices regardless of protocol, manufacturer, or model, and handle mouse clicks the same as well.

The device developer can chose to display a table in either of two directions:

- Vertical (column heading on top)
- Horizontal (row headings to the left)

The device developer can define the heading describing the series of data using the VECTORS that will assist the technician in reading the table.

**Vertical Grid**

By selecting VERTICAL, the device developer defines that the data heading will be shown on top for the columns running from left over to the right.

Found Echoes		
Type	Distance	Amplitude
Surface echo	0.368	4736
Unknown echo	0.698	1913
Unknown echo	0.950	1089
Unknown echo	1.133	827
Unknown echo	1.481	405

**Figure 29** Table with headings for three columns

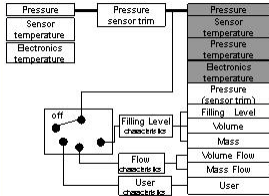
The device developer uses HANDLING to control if the technician only can read the data in the grid or if the values can be edited.



language selected by the technician. A better way may be to consider separate multi-lingual text and use of the standard dictionary to ensure correct language.

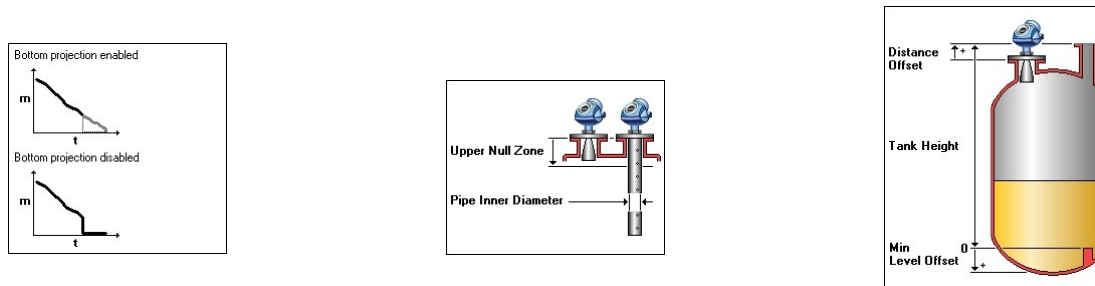
Keep in mind that device management software is not a process flow mimic operator screen. Instead of using dozens of conditional images to emulate level, just use the readily provided bargraph.

The device developer can include block diagrams of hardware and software to assist the technician in the understanding of the internal workings of the device, which may facilitate faster configuration and troubleshooting.



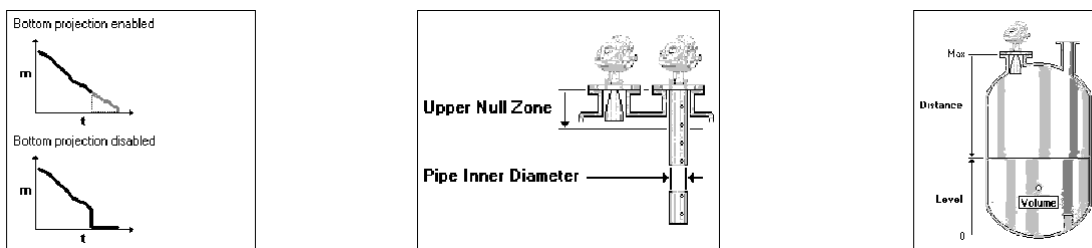
**Figure 34 Image illustrating internal signal flow in DP transmitter**

The device developer can include images illustrating different tank shapes and how the dimensions are taken for use in volume calculations.

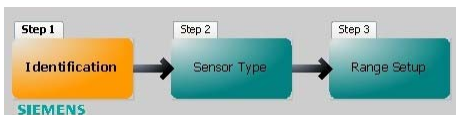


**Figure 35 Image illustrating advanced concepts**

Images are very flexible and are limited only by the device developer's imagination. The device developer can include images depicting timing diagrams for timers and pulse trains for resolves, as well as logic diagrams and truth tables - anything that can assist the technician in understanding of advanced concepts.

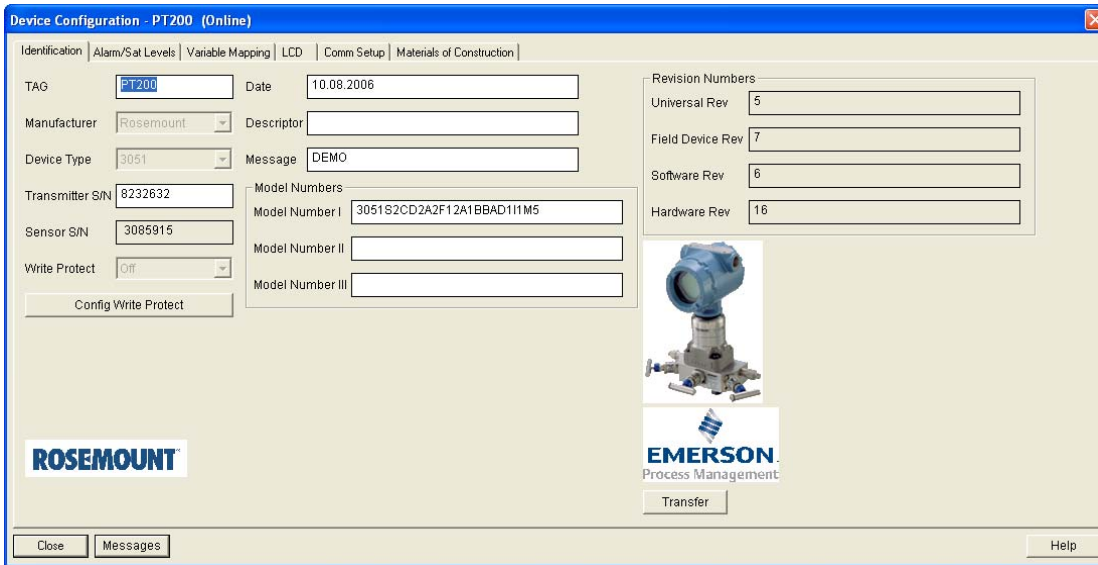


**Figure 36 Image illustrating advanced concepts as it appears in a simple handheld communicator**

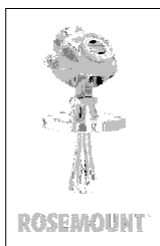


**Figure 37 Image illustrating progress in advanced setup wizard**

The device developer can include company logo on the identification page to support strong product branding and corporate identity. Contact details can be added as static text.

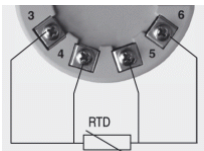


**Figure 38 Company logos and product photo in identification page**



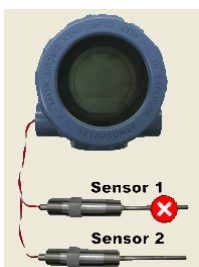
**Figure 39 Company logo and product photo in simple handheld communicator**

The device developer may also include images illustrating correct mounting positions and proper wiring of the device, thus assisting the technician in any replacement or commissioning work. The device developer can use conditionals to switch the image depending on previous settings or current status. For example, to illustrate sensor wiring depending on sensor selection.



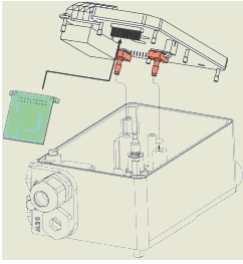
**Figure 40 Image illustrating wiring**

The device developer may also include images illustrating the fault, thus assisting the technician troubleshoot. The device developer can use conditionals to switch the image depending on status. For example, to illustrate a fault depending on status. That is, conditional images can be used to make the page animated.



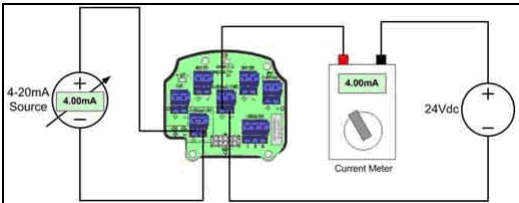
**Figure 41 Image showing fault**

Images can be used to guide technician to internal components



**Figure 42 Exploded view**

Images can be used to illustrate wiring of test equipment

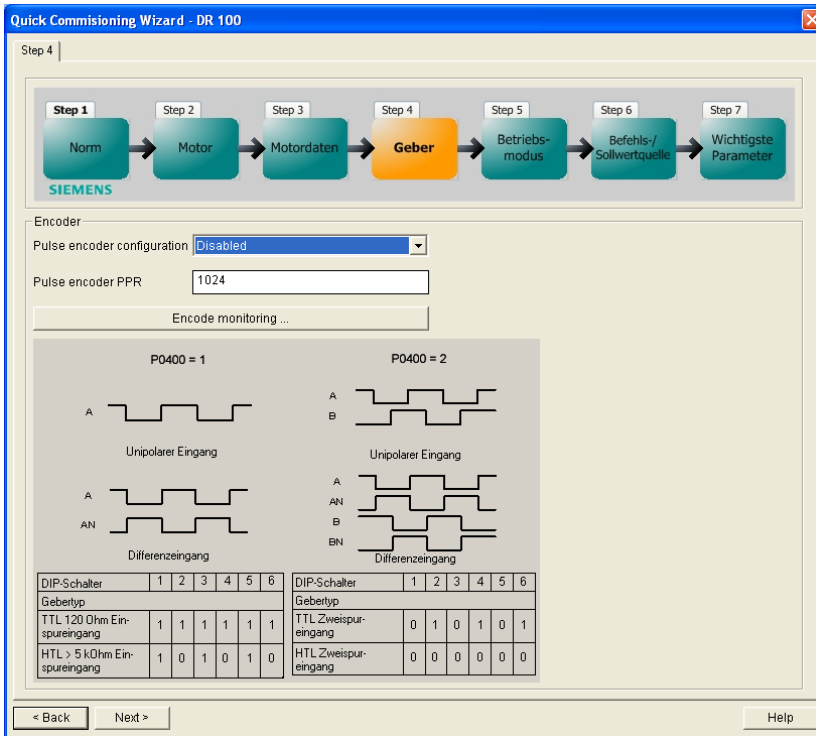


**Figure 43 Wiring diagram**

The device developer can add action to the image in such a way that when technician clicks on the image it opens a menu (e.g. dialogue box or window) or invokes a method.

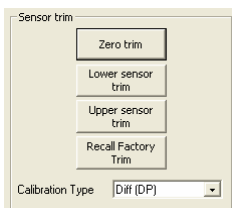
### ***METHOD: Wizards and Scripts***

Method scripting has existed since 1992 but is taking on a larger role as a result of the enhancements in IEC 61804-3. Methods embedded in EDDL are interpreted to make device displays dynamic and interactive just like JavaScript embedded in HTML is interpreted to make web pages dynamic and interactive. The visual enhancements have made EDDL applicable to new and more sophisticated devices. Device developers can use methods to create wizards that take the technician through complex setup procedures by only displaying the valid options depending on earlier selection and ensuring all necessary settings are done. The wizard enforces proper sequence for procedures. Methods are also used to deal with the internal "rules" of the device, such as dependencies between parameters. Wizards hide the complexity of fieldbus devices by isolating the technician from the many steps associated with setting mode in block before performing procedures such as calibration or setup. With the new device-level access (cross-block), wizards can ensure consistency between range configuration in function blocks and transducer blocks. Thus methods demystify complex devices and bus technologies. For instance, for FOUNDATION fieldbus parameters that require mode to be changed before changing online, provide a wizard to handle it automatically for the user. The wizard procedure flow can branch into different steps depending on prior selections. One wizard can start the next, for instance calibration after setup is completed.

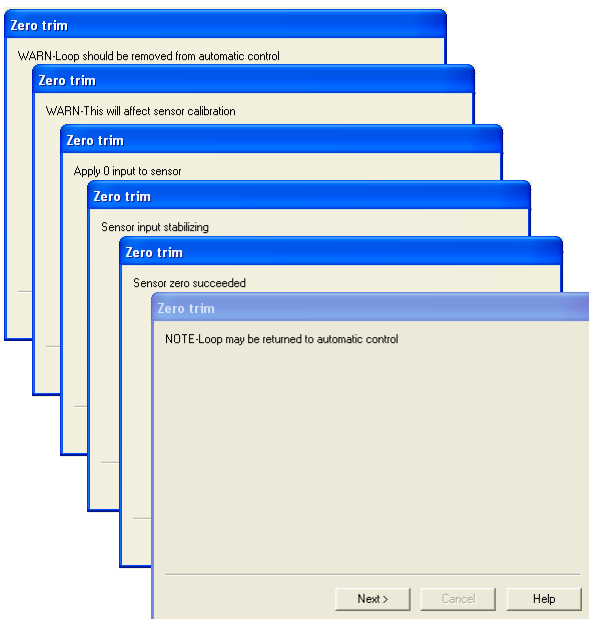


**Figure 44** Advanced setup wizard for variable speed drive

Similarly the device developer can use methods to simplify and eliminate mistakes in complex calibration trim procedures, including reminders to ask operations to “put loop in manual” in order to prevent disturbing the process.



**Figure 45** Methods are invoked at the click of a button



**Figure 46** Methods takes user step-by-step through complex calibration trim procedures

Method can be used to set default values or typical application settings.

Device developers can also use methods for complex interaction with a device as may be required when retrieving large amounts of data, such as waveforms, or retrieving data from the device management software's database. This includes ensuring the device is in the right mode before reading and writing.

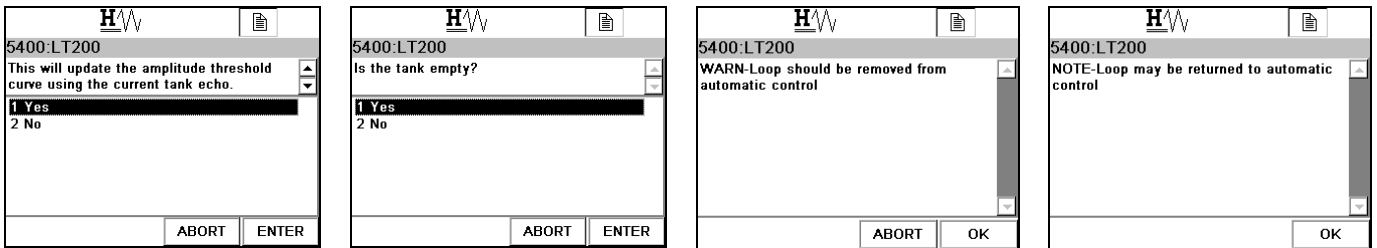


Figure 47 Methods takes user step-by-step through complex setup procedures

### HELP: Context sensitive guidance

Help has existed since 1992 but has been extended to the new graphical elements in IEC 61804-3. The device developer can provide help extensively throughout device descriptions. Help, to be useful, must be context sensitive and therefore the device developer should provide help for every variable and block, every option, every menu (window, dialog box, tab, and frame), every image, and every method. Device developer should also provide help for graphs and charts as well as their individual sources, waveforms and axes. This makes devices easier to use. The device management software displays help the same way for all devices regardless of protocol, manufacturer, or model. The user need not have to search a large chunk of text.

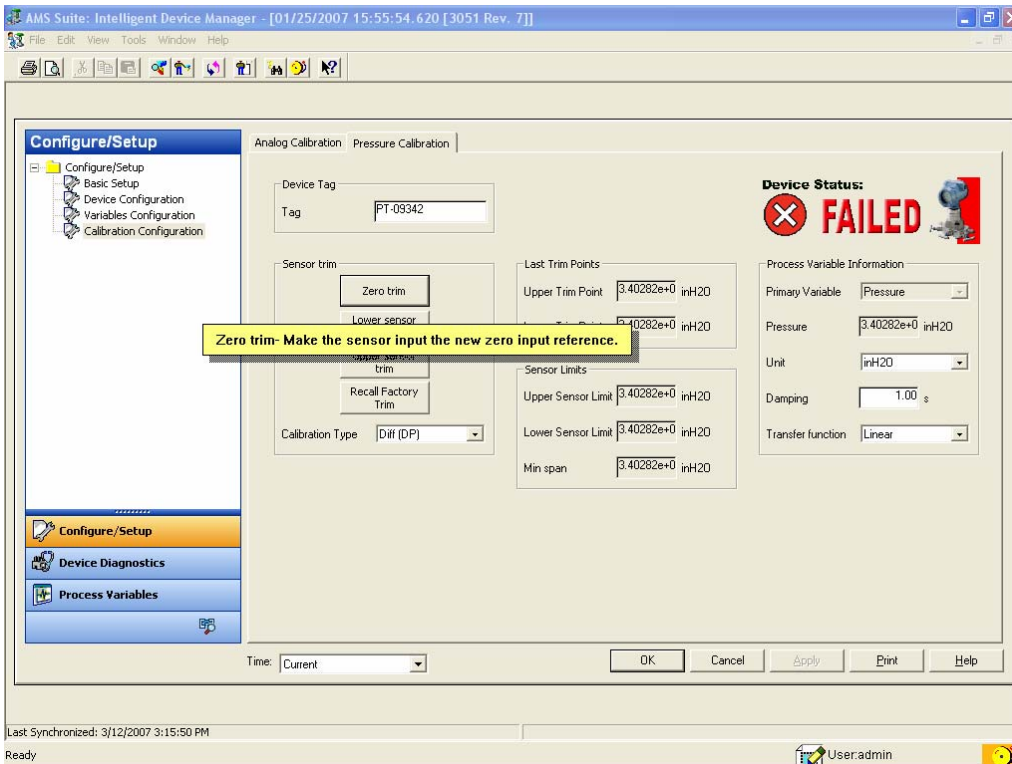


Figure 48 Context sensitive help is provided by the device developer

### FOUNDATION fieldbus device-level access

A new feature in version 5.1 of the EDDL profile for FOUNDATION fieldbus is the device-level access (cross-block) functionality that allows data from multiple blocks to be combined into the same screen. That is, data from the resource and multiple transducer blocks and function blocks can be shown together. This also has the benefit that blocks disappear from the menu system. That is, the user need not know in which block the parameters associated with a particular function or task are located. This also means that FOUNDATION fieldbus devices appear very much the same as non-

block-oriented protocol devices such as HART instruments. Work is done at the device-level, not the block-level.

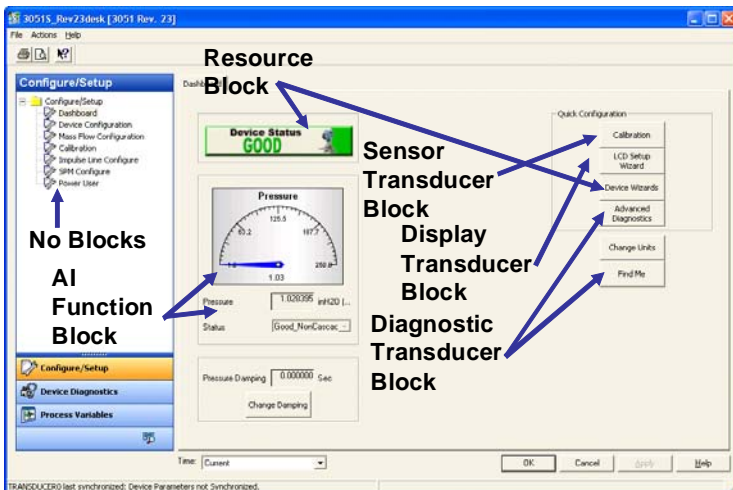


Figure 49 Device-level access support removes block segregation (Software not yet released) Dictionary

## Dictionary

To ensure even greater consistency in display and user interaction, the device developer can make use of the multilingual dictionary built into EDDL, which includes frequently used help text, parameter labels, error messages, and user prompts. Moreover, the device developer has less work to achieve multilingual operation. For example, the standard phrase "Do you wish to zero the input and try again?" automatically becomes "Soll Null angelegt und neu versucht werden?" if user has selected German language in the device management software.

## Engineering Software

Using EDDL, Profibus engineering software can eliminate the need for integrators to know slot/index of the cyclic and acyclic parameters in the device.

## Implementation

EDDL allows the device developer to display the device just the way they want it. Intelligent devices have not been easy to use in the past. Many parameters were not used. Here is a way to make them easier for the technicians to use.

EDDL does not change the device; it does not require any changes to device firmware. EDDL is a file that exists in a computer or handheld without affecting the device. Therefore, EDDL can be written for existing HART, FOUNDATION fieldbus, and PROFIBUS devices. The device developer declares the display using the EDDL markup language. It is like designing a web page using HTML keywords to anchor images and controls on the display. The resulting EDDL file is independent of operating system, service packs, device management software, CPU, and screen size. Programming is not required; third-party components and controls are not required either. Most devices already have EDDL files, which now only need some simple updates to take advantage of the new graphical capabilities. Variable declaration and communication are unaffected. Making the transition is easy. The IEC 61804-4 style guide contains EDDL source text examples to help get you started.

Each version of the device has one EDDL file that only deals with that version of the device. There is no need to incorporate more and more functionality within one file to cover exception and special conditions for every new version of the device with the consequence of reduced robustness and increased effort for development, test and maintenance.

EDDL is based on individual files for each device type, enabling timely release of a new device as soon as it is ready, no need to wait for entire family. Similarly, if an improved user interface is



So what can't the device developer do with EDDL? To ensure a consistent user interface across the vast assortment of devices used within a given facility, the device developer can only control the contents and structure of the display, not the details of the "look & feel". For example, the device developer cannot control:

- Chart background color and grid size
- Color of pens/waveforms in charts and graphs (can be overridden but not recommended)
- If needle gauge is 180 degrees or 270 degrees, and its background color
- Graph background color and grid size
- What the button icons for pan/zoom/home look like and their location
- What the button icons for help and print look like and their location
- If there should be a header or not
- If there should be a navigation tree or not
- How its indicated that changes have been made and need to be downloaded to the device
- Language selection
- If there should be a status bar or not, and its location
- What the button icons for sending changes to device look like and its location
- Display background color
- Font, color, and size for text
- Where to click to open the manual

This ensures that all of this is exactly the same for every device in the system, regardless of protocol, manufacturer, or model.

With EDDL enhancements device developers can now design the kind of system display they want but previously were unable to. EDDL are low maintenance, they need not be updated as new Windows versions are released. The system display will continue to work even if customer upgrades Windows on their system.

There are companies that specialize in helping device manufacturers write EDDL files for their devices.

### ***EDDL IDE and Developer Workshops***

Integrated Development Environment (IDE) software are available to all device developers to write, test, and debug EDDL files for devices all in one application. The IDE features method debugger and customizable tag files for quick code completion and customizable color-coding as the developer types.

The developer can use the method debugger feature of the IDE to help test and debug the methods. Breakpoints can be set within the methods to determine if the methods are operating properly in an incremental fashion. The value of variables and parameters can be seen in the watch window. The developer can see charts and graphs in the viewer window in real time, or simulation files can be used for testing. The developer can also view menus in the IDE. They are represented as pull-down menu buttons in the viewer window.

The IDE generates distributable files for old systems without enhancements as well as new systems with the graphics capability, thus ensuring backwards compatibility.

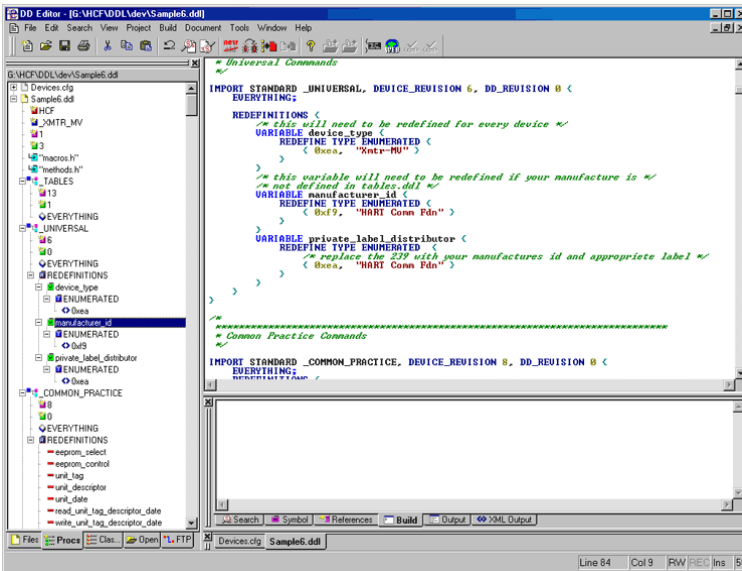


Figure 51 Integrated development environment for EDDL

The IDE allows the device developer to do more complete testing, to simulate user interaction, and results in improved quality assurance for the product and therefore a better user experience.

Protocol	EDDL IDE Suppliers	URL
HART	HCF	<a href="http://www.hartcomm.org">www.hartcomm.org</a>
FOUNDATION fieldbus	FF	<a href="http://www.fieldbus.org">www.fieldbus.org</a>
PROFIBUS	Ifak	<a href="http://www.ifak.eu">www.ifak.eu</a>

The software suppliers also organize EDDL development workshops where device developers can learn how to write EDDL files for devices.

### Future devices

For the future, device developers may consider including new value added features such as diagnostics and performance analysis to devices to take full advantage of EDDL capabilities. EDDL is what enables device developers to innovate and add new parameters, commands, and blocks, to make their devices better yet ensuring they remain compatible with other systems, and to integrate their devices into control systems without having to install any software.

### Conclusion

Device developers should add better menus and graphics to their existing DD files. Graphics and menus created for one device can typically be reused for other devices as well minimizing the effort and promoting consistency. Device developers are in complete control of content and structure (e.g. if there is a button or not, and where in the menu the button is located) of the display of the information about the device. The DCS only determines the look & feel (e.g. the size and color of the buttons) the DCS does not determine the content and structure. Device developers should use all the capabilities of EDDL technology to display all the information in the device, and methods for calibration and setup wizards. EDDL makes advanced diagnostics easy to use. Customers who would not have considered investing in advanced device diagnostics before out of fear not being able to set it up or use it, will dare to do so now.

### References

IEC 61804-3 Ed. 1.0 English, Function blocks (FB) for process control - Part 3: Electronic Device Description Language (EDDL)

IEC/TR 61804-4 Ed. 1.0 English, Function blocks (FB) for process control - Part 4: EDD interoperability guideline

EDDL Brochure and Technical Description on [www.eddl.org](http://www.eddl.org) site

Jonas Berge, "Fieldbuses for Process Control: Engineering, Operation, and Maintenance", ISA, 2002, ISBN 1-55617-760-7