

**FieldCommander<sup>®</sup>**

**User's Guide**

**FC-SW**

# Table of Contents

<b>About this manual</b> .....	<b>5</b>
<b>1. Overview of FieldCommander</b> .....	<b>6</b>
1.1 Introduction .....	6
1.2 Scripting over system languages .....	6
1.3 Key features .....	7
1.4 A quick overview .....	7
<b>2. Creating an application</b> .....	<b>10</b>
2.1 The requirements first .....	10
2.2 Making a design .....	10
2.3 Finally the implementation .....	12
<b>3. System configuration</b> .....	<b>15</b>
3.1 Introduction .....	15
3.1.1 Customizing the pages .....	15
3.1.2 User accounts .....	15
3.2 Opening the configuration .....	15
3.2.1 The home page .....	15
3.2.2 Logging in .....	16
3.3 Script setup .....	18
3.4 System settings .....	19
3.4.1 IP addressing .....	20
3.4.2 DNS lookup .....	21
3.4.3 Time settings .....	22
3.4.4 Mail server .....	23
3.4.5 FTP server .....	24
3.4.6 Services .....	24
3.4.7 Logs .....	26
3.4.8 SSL certificate .....	27
3.4.9 System information .....	28
3.5 User profiles .....	28
3.6 Database admin .....	30
3.6.1 Databases .....	30
3.6.2 Tables .....	31
3.6.3 Table data editor .....	32
3.6.4 Edit table columns .....	33
<b>4. System administration</b> .....	<b>34</b>
4.1 File management .....	34
4.1.1 Connecting with FTP .....	34
4.1.2 Installing your applications .....	35
4.2 Working with scripts .....	36
4.3 Log files .....	37
4.3.1 FCscript logging .....	37
4.3.2 PHP logging .....	37
4.4 Security .....	37
4.4.1 Passwords and profiles .....	37
4.4.2 Secure web server .....	38
4.4.3 Firewall configuration .....	38

---

4.5	Rebooting the system	38
4.6	Updating the software	39
<b>5.</b>	<b>Data sharing</b>	<b>40</b>
5.1	Data buffers	40
5.1.1	Accessing data	40
5.2	Meta tags	41
5.2.1	Using meta tags	41
5.2.2	System tags	42
5.3	Databases	43
5.3.1	Creating and editing	43
5.3.2	Permanent storage	43
5.3.3	Accessing the tables	43
5.4	Data exchange	46
<b>6.</b>	<b>Application script features</b>	<b>47</b>
6.1	Data processing	47
6.1.1	The flow of information	47
6.1.2	Components	48
6.1.3	Conditions and expressions	52
6.2	Event handling	53
6.2.1	Timer events	53
6.2.2	Event callbacks	53
6.3	Features	54
6.3.1	Data from streams	54
6.3.2	Data exchange over network	54
6.3.3	Telephony	56
6.3.4	Audio	56
6.3.5	Imaging	57
<b>7.</b>	<b>User interface features</b>	<b>58</b>
7.1	Dynamic web pages	58
7.1.1	SSI meta tags	58
7.1.2	PHP engine	58
7.2	Real-time web page updates	59
7.3	Graphical user interface	61
7.3.1	Real-time webbased presentation	61
7.3.2	WebGUI in a web page	62
7.3.3	WebGUI configuration	63
<b>Appendix A: RS-232 communication</b>		<b>66</b>
<b>Appendix B: RS-485 communication</b>		<b>69</b>
<b>Appendix C: Software editions</b>		<b>71</b>
<b>Appendix D: SQL syntax</b>		<b>72</b>
<b>Appendix E: WebGUI components</b>		<b>75</b>
<b>Glossary</b>		<b>86</b>
<b>Index</b>		<b>89</b>
<b>Notice to the user</b>		<b>91</b>

---

Software License Agreement . . . . .	91
Third-party software . . . . .	92
Trademarks . . . . .	93
Copyrights . . . . .	93

## About this manual

Congratulations on your purchase of FieldCommander!

This document focuses on FieldCommander from a user's perspective and explains all its features, how to setup, operate and manage the device and how to create and employ your applications.

### Organisation of the documentation

#### Installation

*Installation Guide Hardware Platform 1 (FCHWP1IG)*

*Installation Guide Hardware Platform 2 (FCHWP2IG)*

Bundles information related to the hardware, such as the installation of the device, IP networking, serial interface details and technical specifications.

#### Employment

*User's Guide (FCSWUG)*

Explains how FieldCommander is used in your application development, and documents all its features, including the databases, communication and web based configuration.

#### Programming

*FCscript Programmer's Guide (FCSCRIPTPG)*

Function reference of FieldCommander's scripting language, used to control the data flow and event management in your applications.

*FCphp Programmer's Guide (FCPHPPG)*

Function descriptions of the FieldCommander specific PHP interface, used to build dynamic user interfaces or reports on top of the your application.

*Javascript Reference Manual (FCJSREF)*

Extensive documentation of the Javascript (ECMAScript) core language used by FieldCommander for application programming.

#### Software options

*Modbus Option Guide (FCOPT10OG)*

*Galaxy Option Guide (FCOPT20OG)*

*EIB/KNX Option Guide (FCOPT30OG)*

*RDA Option Guide (FCOPT40OG)*

The "plug in" modules (optionally available) have their own documentation to explain the features, installation, configuration and programming interfaces.

# 1. Overview of FieldCommander

## 1.1 Introduction

Devices in the field are often interconnected with industry standard interfaces. Although the protocols are 'open' (publicly available), the systems and applications they operate in are mostly closed. This means features cannot be added or used to interface with other systems or applications.

FieldCommander puts you in control of the devices on the field bus and bridges the gap to the office environment. The use of Ethernet and an extensive set of supported TCP/IP services, such as the built-in Web server and FTP server, allow you to access and manage the devices over local area networks and even the Internet.

Interpreting and analyzing data by the execution of custom scripts are at the very heart of FieldCommander. You can set up conditions to be handled by the event manager according to your needs. The information gathered can be reported by file or e-mail and visually presented through a custom web pages, to name a few options.

## 1.2 Scripting over system languages

Some advantages of scripting, compared to system-level languages:

- **Faster development**  
Scripts take much less time to write than system code. The programmer must do less housecleaning as the script platform handles most of the tedious memory and data problems.
- **Safer**  
The script environment protects against programming errors, for example dividing by zero would crash most systems, but is caught by the scripting system. All calls in a script environment must pass through the script environment, so potentially dangerous calls are prevented.
- **Flexible gluing**  
Ideal scripting solutions glue existing components together. FieldCommander excels at this point as it features a large set of high-level commands, not found in any other product.
- **Smaller code**  
A single line of scripting code often accomplishes many lines of system-level code. Less code means less errors.
- **Faster to market, smaller, cheaper**  
FCscript and FCphp together allow you to develop significantly faster so you can employ your application earlier, for less money.

## 1.3 Key features

All FieldCommanders share the following main features:

- Event based processing to timely act on situations
- Serial ports to interface with different equipment over RS-232 and RS-485 lines
- Relational SQL database to store application and configuration information
- Visualization and configuration using a standard and secure web browser
- Build your own user interface with dynamic web pages using PHP scripting language
- User profiles with authentication for personalized and secured access
- FTP file access to up- and download your application and web pages
- Alert and report over e-mail and text messages (SMS, with optional GSM modem)
- Report telemetry data through GPRS (with optional GSM/GPRS modem)
- Share data over TCP/IP network

The FieldCommander Advanced Software edition adds:

- Real-time web page updates in a web browser through Java™
- Customizable real-time graphical user interface in a web browser through Java™
- Atomic clock synchronisation using external NTP server
- FTP file access from the script to send and receive files in your application
- Imaging features to take pictures (with optional USB camera)
- Sending of multimedia messages (MMS, with optional GSM/GPRS modem)
- Audio and interactive voice response features (on selected models)

The FieldCommander Extended Software edition adds to the Advanced Software:

- Fully integrated web browser with Java™
- Touchscreen with a virtual pop-up keyboard

Optional protocol converters for all software versions add:

- Direct communication with Modbus ASCII devices (FC-OPT10)
- Communication with Honeywell (Ademco/Microtech) Galaxy Alarm panels (FC-OPT20)
- Full interfacing with EIB/KNX networks (FC-OPT30)
- Easy integration of remote data acquisition (RDA) modules (FC-OPT40)
- Communication with wire-less Zigbee networks (FC-OPT50)
- Receiving data from battery-less and wire-less EnOcean sensors (FC-OPT150)
- Integration of ESPA 4.4.4 based pager systems (FC-OPT160)

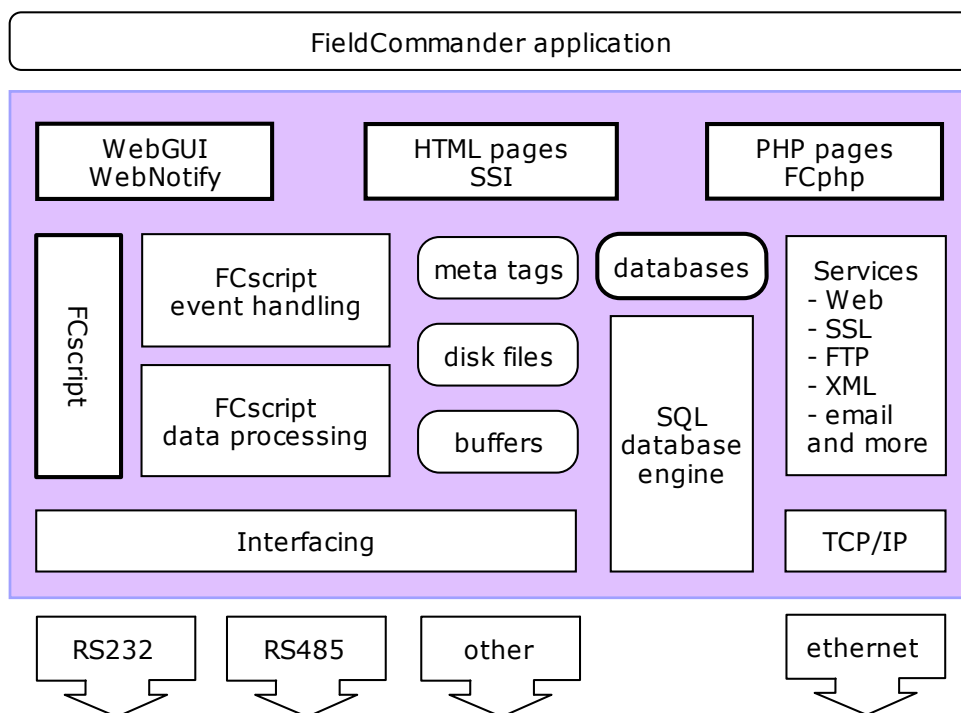
For a complete overview of the various software editions, refer to Appendix C.

## 1.4 A quick overview

FieldCommander has two major terminators; the serial interface and the network interface. Through these interfaces, FieldCommander puts you in control of the serial devices and bridges the gap to the office environment through the network connection.

The serial ports are dedicated to send and receive the data on the RS-232 or RS-485 ports. The event handling provides you with full control over data and enables you to report via e-mail or visualize data on a custom made web page, for example.

The picture below shows the most notable functional blocks of FieldCommander.



### Interfacing

The interface drivers provide the means to connect your serial devices to the data processing of FCscript. FieldCommander supports RS-232 and RS-485 and several other interfaces, depending on the model.

### Data processing in FCscript

FieldCommander features a data processing engine to detect and select the occurrence of data patterns, values, states or times, and includes protocol converters for various popular protocols today. The exact processing of data is set up and configured by your FCscript.

### Event handling in FCscript

Events are situations triggered during data processing, timers or by user input. You can take any responsive action by adding your code in the central callback function *handleEvent()* of your FCscript. A large set of FCscript functions and services is available to do what whatever you like.

### Web based user interface

FieldCommander's user interface is entirely web based and can be completely customized. The system itself is managed through PHP pages (FCphp, source is included). Your own interface can be composed of plain HTML pages, dynamic pages with PHP and even with real-time graphics using the WebGUI applet.

### SQL databases

Your databases can be accessed both from FCscript and FCphp to serve as a powerful way to share structured information. They can be accessed by simple database functions or through true SQL queries.

## Services

The built-in services of FieldCommander include:

- **Web server with SSL**

The web server provides access to the built-in configuration system, and your custom user interface. They can be accessed over a local network or even the Internet via any standard web browser. The web server can also run securely with SSL encryption to protect the traffic from being tapped.

- **FTP server**

You can up- and download files directly to and from FieldCommander's flash disk using an FTP application. Typically, scripts and custom web pages are created/edited locally using your favorite text editor and uploaded to FieldCommander once they are finished. You can control who has access to which parts of the disk through the user settings, or block FTP access completely.

- **Mobile messaging**

With an optional GSM modem, FieldCommander can send and receive text messages known as SMS. This enables you to send alerts and reports *to* mobile phones but you can also receive and interpret instructions sent *from* mobile phones. Besides sending plain text, you can add a picture. This is called Multimedia messaging, or MMS (not on all models).

- **E-mail, FTP client, XML**

Several network based services are built into FieldCommander. You can send text messages with reports or alerts as an e-mail. With the built-in FTP client you can send and receive files directly from a running FCscript, for example to send (log)files to a remote computer. The export of XML files provides a universal data transfer mechanism to share data with other systems or applications.

- **Clock/timer**

A real-time clock provides accurate time and date functions. In addition, FieldCommander supports the NTP protocol. By assigning an NTP server, you can periodically synchronize FieldCommander with an external time source.

## TCP/IP interface

Network connections to and from FieldCommander are established through the proven TCP/IP protocol stack via a 10/100 Mbit RJ-45 Ethernet port. This provides maximum flexibility and usability in today's computer networks.

## 2. Creating an application

You can let FieldCommander do virtually anything you want, but you'll have to give it instructions first. With an application at hand, you will define the requirements in terms of data to be processed, situations (events) to handle and information to be presented. This chapter will show the steps to create an application with FieldCommander in a simple tutorial and provides a real-world example.

When developing a script for FieldCommander, you will go through the same steps as you would programming other software. You will see however, that especially the "design" and "implementation" (programming) steps are very straight forward. It will cut back development time to a few hours as compared to the weeks or months a third-generation general purpose language usually takes to develop.

### 2.1 The requirements first

First we need to collect the requirements and constraints for the application. In this case, the application should (a) show the current measured value in a web page. It should (b) only show positive values and (c) give a visual warning when the measured value exceeds "+9.00".

The measurement device is an analog input module which uses an ASCII-based request/response protocol. Only when the device is polled by sending a request query, it responds with the current value. As these values will not change very quickly, it is sufficient to poll the measurement device every 10 seconds.

A request query starts with a hash (#), followed by the address of the device and ends with a carriage-return character (symbol: `<code>\r</code>` or `<code>\r</code>`, ASCII value 13 decimal):

request example: "#01\r"

The device sends a reply immediately, starting with a greater-than character (>), followed by the measured value and ended by a carriage-return character:

response example: ">+023.20\r"

This protocol is used in distributed data acquisition systems, like for instance Advantech's Adam 4000 module series. It communicates over RS-485 at 9600 bps, 8 data bits, 1 stop bit and no parity. FieldCommander simulates such modules on port 0 for demonstration purposes.

### 2.2 Making a design

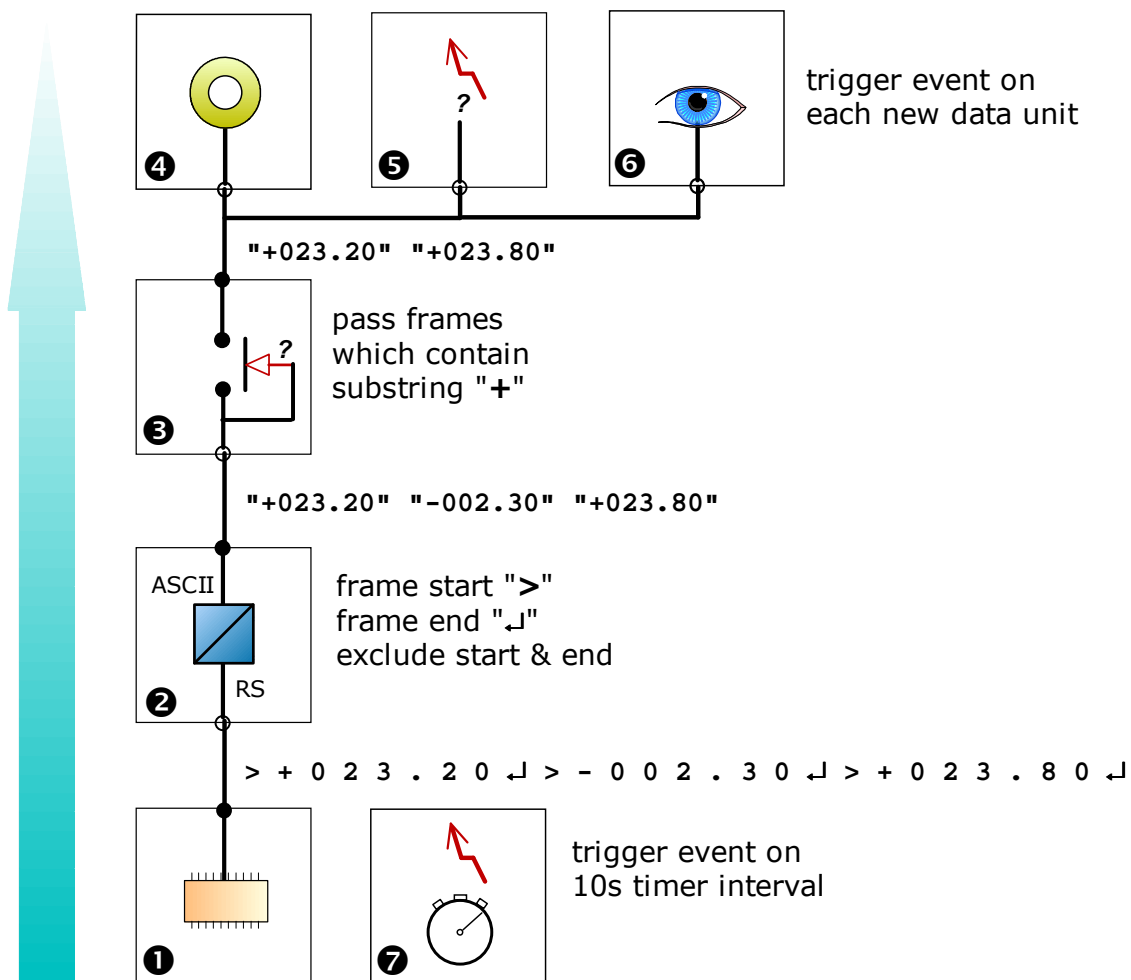
Given the requirements of the application, a list can be made of components which are necessary to achieve the goals.

Communication with the measurement device is handled by a serial port of FieldCommander, called Local Port . The responses of the device are acquired. Every character of the response query is put into a separate data unit (DT\_RS), each with time stamp and control signals.

Because all characters are in separate data units and there is no 'framing' and no beginning nor end to the stream it is difficult to interpret the data stream. This is where the ASCII Frame Converter comes in. Linked to the Local Port, it reads all data units in the stream, detects the start and end of each frame and converts them into a new data unit type (DT\_AF). This data unit type contains the measurement value (between the > and ↵ characters) as a single string.

Now that we have the measurement value in a single data unit, it is time to pass it to the web page. To insert dynamic content into a web page, a meta tag must be set in the meta table (requirement a). This is done by the event manager, which requires a Buffer to temporarily store the data units, and a DataTrigger to generate an event. But we should not forget to Filter out the negative values first (requirement b). The Viewpoint is used to pass the values in the stream to the graphical user interface.

Let's illustrate it with a diagram of the network. This proves to be very helpful while making designs. After all, a picture is worth more than a thousand words.



Every data unit going through the Filter component will cause the DataTrigger to generate an event, which is processed by the event manager. On the occurrence of an event, FC needs to read the measurement value of the new data unit from the buffer and put into several meta tags, so the latest reading is send to the graphical user interface (WebGUI).

Now that we have our hands on the value, it is easy to check if it exceeds "+9.00". The result is stored in other meta tags so it can be indicated on the WebGUI (requirement c).

In the example we use a `Timer` to generate an event on a 1 second interval. We will use this to transmit the request query to the measurement device every second.

## 2.3 Finally the implementation

A script file typically contains two functions; `main()` and `handleEvent()`. The main function is executed when the script is started. Here we put all function calls to configure the port(s), set up the components in the stream(s), initialize the defaults for the application and start the data processing. To prevent the function from returning (which will cause the application script to stop) we put the script to sleep.

The `main()` function of the script file (example1.jse) is printed below:

```
file: example1.jse
1 //
2 // example1.jse - demonstrates a typical script setup
3 //
4 // For information see accompanying FieldCommander documentation
5 //
6
7 function main() {
8
9     // Open and configure simulation port 0
10    demoport = newLocalPort(0, DT_RS);
11
12    // Setup data processing
13    positive = setStringCondition(DT_AF, "data", "+", SUB_STRING);
14    convert = addConverter(demoport, ASCII_FRAME, ">", "\x0d", 0);
15    filter = addFilter(convert, "positive", REPEAT);
16    buffer = addBuffer(filter, "mybuffer", 1000, RING);
17    newdata = addDataTrigger(filter, "", REPEAT);
18    vpoint = addViewpoint(filter, "measureview");
19
20    // Initialize interval timer every 10 seconds
21    polltime = newTimer();
22    interval.sec = 10;
23    interval.usec = 0;
24    startIntervalTimer(polltime, interval);
25
26    // Initialize meta tags
27    warnmode = 0;
28    totalled = 0;
29    setMetaValue("totalinled", "0");
30    setMetaValue("totalmeter", "0");
31    setMetaValue("warning", "0");
32    currentDate = new Date();
33    setMetaValue("warnmsg", "Alarm reset at "+currentDate.toLocaleString());
34    delete currentDate;
35
36    // Start and go to sleep, script will be event driven
37    openPorts();
38    sleep();
39 }
```

The design is easily translated into a script. Line 10 initiates the local port to which the measurement device is connected. In this example it is simulated by port 0. Line 14 to 18 configure the components , , , and and link them up the network. Note how the filter (line 15) uses a simple expression on the string condition defined in line 13. The timer is set up and started by lines 21 to 24. To make sure the meta tags exist before an event is triggered, they are initiated with defaults in line 29 and 34.

Now that the data processing is set up, the ports can be opened (line 37). This is necessary to make the serial ports accept and process data. To avoid ending the script, we put it to sleep indefinitely in the last line (line number 38). This makes the script completely event driven.

The script handles two events, both of which are set up in the network: the DataTrigger "newdata" (line 17) which is generated every time a data unit passes the filter and enters the buffer, and the 1 second interval timer "polltime" (line 21 to 24).

The second part of the script, the `handleEvent()` function is printed here:

```
file: example1.jse
40 //
41 // The event callbacks are managed here
42 //
43
44 function handleEvent(event, type) {
45
46     switch ( event ) {
47
48     case polltime:
49         sendString(0, "#01\x0d");
50         break;
51
52     case newdata:
53         gotoNext(buffer);
54         result = getBufferDataElement(buffer, DT_AF, "data");
55
56         totalled +=parseFloat(result);
57         setMetaValue("totalinled", totalled);
58         setMetaValue("totalmeter", totalled % 100);
59
60         if ( !warnmode && ( parseFloat(result) > 9.0 ) ) {
61             warnmode = 1;
62             setMetaValue("warning", "1");
63             currentDate = new Date();
64             setMetaValue("warnmsg","Alarm HIGH at "+currentDate.toLocaleString());
65             delete currentDate;
66         }
67         break;
68
69     case 1: // trigger from hotrect in applet
70         if ( warnmode && (type == evApplet) ) {
71             warnmode=0;
72             setMetaValue("warning", "0");
73             currentDate = new Date();
74             setMetaValue("warnmsg","Alarm reset at "+currentDate.toLocaleString());
75             delete currentDate;
76         }
77         break;
78     }
79 }
```

When an event occurs while the script is 'sleeping', the `handleEvent()` function (line 44) is called with 'event' and 'type' arguments. We use a `switch()` construction to process the

different events. When the event source is "polltime" (line 48), the request query is send to the local port.

In case it is the DataTrigger "newdata" which causes the event, there is new data in the buffer. The most recent value is read from the data unit, converted and placed into the meta tags "totalinled" and "totalmeter" (line 50 to 52). Line 60 to 66 check if the value was above "+9.00" and set the "warning" and "warnmsg" tag accordingly.

When one of the meta tags is changed the new value is immediately passed to the WebGUI applet and the value is passed to the components that use the meta tags as shown in the following code snippet of the XML configuration file for the WebGUI.

```
file: gui.xml
100 <component type="image">
101   <name>resetbutton</name>
102   <x-pos>400</x-pos>
103   <y-pos>272</y-pos>
104   <width>101</width>
105   <height>15</height>
106   <visible type="variable">warningmeta</visible>
107   <image value="0">/alarm.png</image>
108   <value>0</value>
109 </component>
110
111 <component type="hotrect">
112   <name>resetalarm</name>
113   <x-pos>400</x-pos>
114   <y-pos>272</y-pos>
115   <width>101</width>
116   <height>15</height>
117   <active type="variable">warningmeta</active>
118   <action>trigger</action>
119   <left>1</left>
120   <right>2</right>
121 </component>
122
123 <variable type="metatag">
124   <variable_name>warningmeta</variable_name>
125   <source_name>warning</source_name>
126   <variable_default>0</variable_default>
127 </variable>
```

This part of the XML file shows how the "reset" button is created to reset the alarm condition. The button is only shown when there is, or has been, an alarm. The first declaration (line 100 to 109) is the picture of the button. When the value of "warningmeta" variable is 0, the image is not visible. The value of the "warningmeta" variable is 1, the image is shown on the given location. The variable "warningmeta" is linked to the meta tag "warning" as shown in line 123 to 127.

To make the button "clickable", a so-called hotrect is placed 'over' the image. The hotrect (as declared on line 111 to 121) is not visible, except that the mouse cursor changes from an arrow to a hand. An event with value "1" is sent to FieldCommander where it is processed by the handleEvent() function as shown on line 69 to 77 of the script. The data processing and event handling goes on indefinitely. It is only stopped when the script is stopped manually from the web-based Configuration System, or by powering off the FieldCommander unit. In the latter case, the script will restart when power is restored.

## 3. System configuration

### 3.1 Introduction

The web interface is the only part of FieldCommander you can actually "see" once it is installed. As it will typically be placed in a meter cupboard or in other unaccessible areas, it doesn't feature controls or a display, except for the reset button and a few LEDs. For information regarding to the external controls and connections, check the appropriate *Installation Guide*.

You can access FieldCommander only by accessing the web interface with a browser or with an FTP client. The web based system configuration provides all user configurable settings and meaningful status information.

#### 3.1.1 Customizing the pages

Everything you'll see in this chapter is based on PHP pages, using the FCphp API. The functions provide access to the system settings, user profiles and databases, and are documented in the *FCphp Programmer's Guide*. The source of the pages shown in this chapter is available through the FTP interface so you can use them as a template to build your own custom user interface. See chapter 4 for details.

#### 3.1.2 User accounts

To control access to the resources of FieldCommander, it uses user accounts also known as user profiles. A profile is protected by a user name and password, and stores access rights to various parts of the disk (over FTP) and web server. New users can be added and the permissions can be changed from the Users tab.

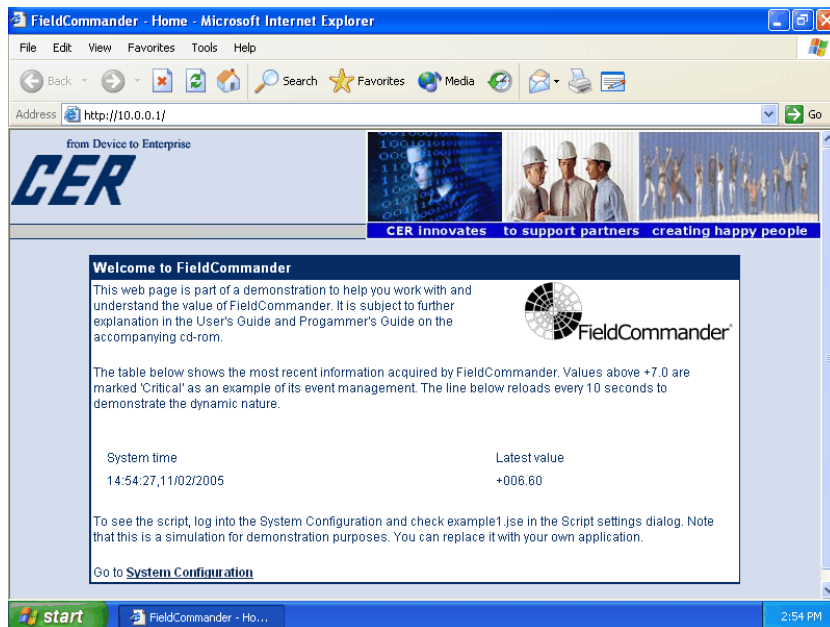
You must log in into the configuration pages with one of the user profiles. The password will be checked and, when successful, a PHP session is created with the logged in user's rights. Some of the configuration tabs in the dialogs can be hidden, depending on the user's rights.

## 3.2 Opening the configuration

### 3.2.1 The home page

To open the configuration pages, start your web browser and point to the IP address of FieldCommander (factory default is 10.0.0.1). This will open index.html. When you have not replaced it by your custom page, it will show the example below.

In case you can't connect to FieldCommander or no page loads, you'll have to reconfirm that the IP address is correct and your network is properly functioning. For detailed information on how to set up the network and IP addressing, refer to the *FieldCommander Installation Guide*.



### 3.2.2 Logging in

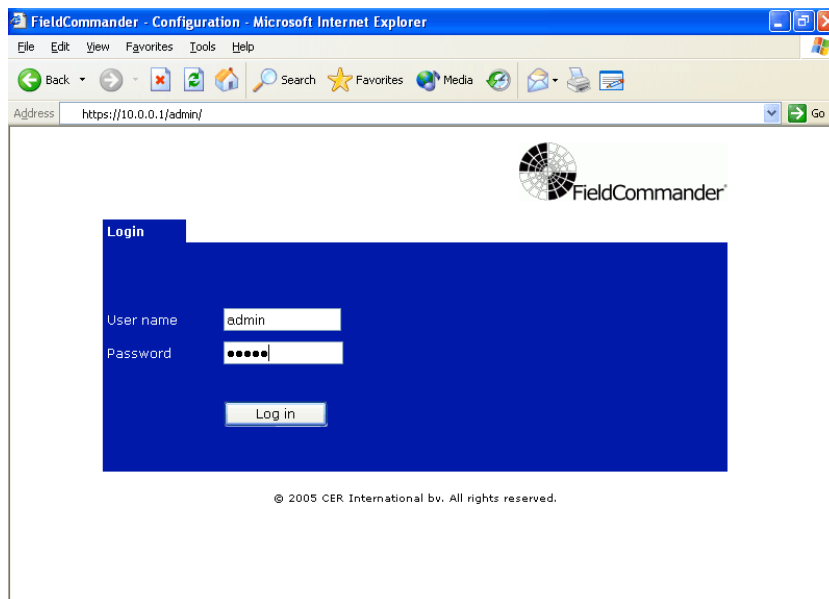
To go the system configuration web pages, follow the "System Configuration" link in the example page or point your browser to the /admin directory of the web server. When FieldCommander is on the default address, this is: **http://10.0.0.1/admin/**

You will be redirected and taken to the login screen. Because it is located at the secure web server, the connection will be encrypted and your browser requests the security certificate from FieldCommander. Because it doesn't know FieldCommander the first time, you will be presented a dialog like this:



You can accept the certificate when you trust it and continue browsing. In case you don't want to accept it every time you access the secure web server, you should add it to your browser's list of trusted certificate authorities.

Now you'll see this login screen:



Here you should identify yourself before you can access any of the configuration pages. Fill out your username and password in the corresponding input fields and Click the "Log in" button. Note that both fields are case sensitive.

Two accounts are preconfigured:

User name: admin  
Password: admin

User profile "admin" has all privileges. This account cannot be altered or removed.

User name: guest  
Password: guest

User "guest" has limited access. You can edit its permissions or delete this account.

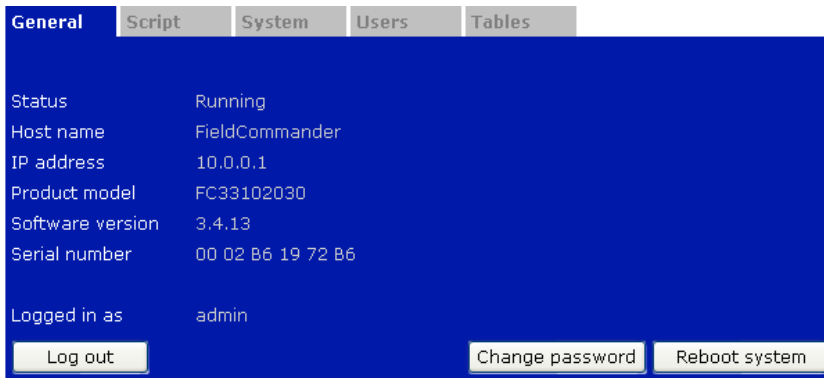
You can create your own user profiles through the User profile configuration (see 3.5).

**Note:** For security reasons, it is strongly recommended to change the passwords.

It is vital to keep the password for the "admin" account in a safe place! If you lose the password, and there is no other account with all admin rights, you cannot regain full control of your FieldCommander.

In case a different page shows, the original configuration page is replaced by a custom version. You can still access the original configurational pages by pointing your browser to **https://10.0.0.1/\_admin/** assuming the IP address is 10.0.0.1. Note the **s** in "https" and the underscore ( **\_** ) prefix in "\_admin".

When logged in, the "general" page of the configuration shows a status overview of this FieldCommander, similar to this picture:



Status:	indicates whether or not an FCscript is running
Host name:	network name of the unit
IP address:	IP network address assigned to this unit
Product model:	the product type identification of the unit
Software version:	currently installed firmware version
Serial number:	unique serial identification also printed on the unit
Logged in as:	name of the user currently logged in

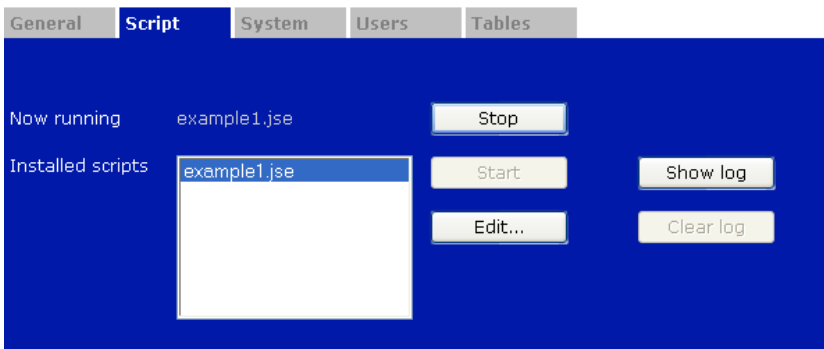
Once logged in, you can log out using the "Log out" button. You will be logged out automatically when you end your browser session by closing all browser windows.

To change the password of the currently logged-in user, press the "Change password" button. You'll have to enter it twice to reduce the chance of typing errors.

The "Reboot system" button is only shown when you have the proper rights in your user profile. Use it to reboot your FieldCommander unit after you have preformed a firmware update. See paragraph 4.6 for details on updating your FieldCommander.

### 3.3 Script setup

From this dialog, you can start, stop and edit the installed script files. Access to these script settings can be blocked/granted per user through the user profiles.



### Start a script

Execute a script by selecting the file name from the list, and click the "Start" button. When there is already a script running, you'll have to stop it first. The selected script is started and will also be executed automatically when FieldCommander is restarted. Check chapter 4.2 to learn how to make your own scripts to be listed here.

### Stop a script

Halt the execution of a script by clicking the "Stop" button.

### Show log

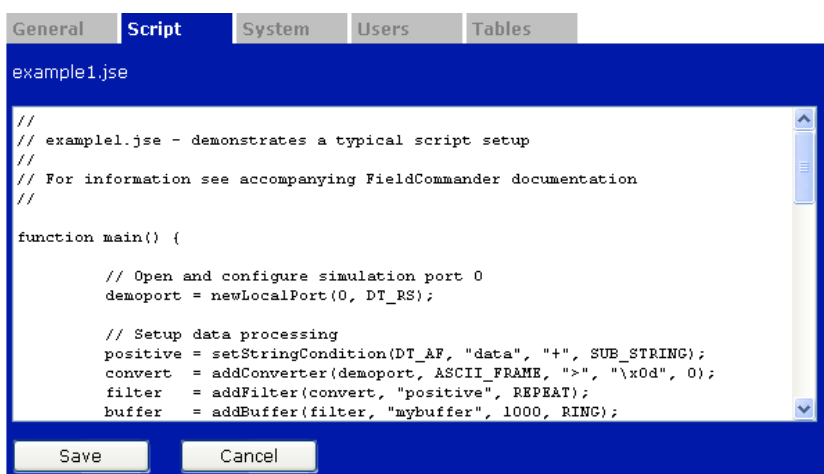
Error and debug messages, as well as information from script is written to a log file during execution of a script. Click "Show log" to view the content of the log file. The reporting level and file size are configured in the System settings.

### Clear log

Erase the current log file. During testing, it can be useful to start with a empty log to easily spot the new entries.

### Edit a script

To edit a script, select the filename and the "Edit" button. The script will appear in an edit box, where changes can be made. Note that a user with "operator mode" rights does not have access to this button.



```
example1.jse

//
// example1.jse - demonstrates a typical script setup
//
// For information see accompanying FieldCommander documentation
//

function main() {

    // Open and configure simulation port 0
    demoport = newLocalPort(0, DT_RS);

    // Setup data processing
    positive = setStringCondition(DT_AF, "data", "+", SUB_STRING);
    convert = addConverter(demoport, ASCII_FRAME, ">", "\x0d", 0);
    filter = addFilter(convert, "positive", REPEAT);
    buffer = addBuffer(filter, "mybuffer", 1000, RING);
}
```

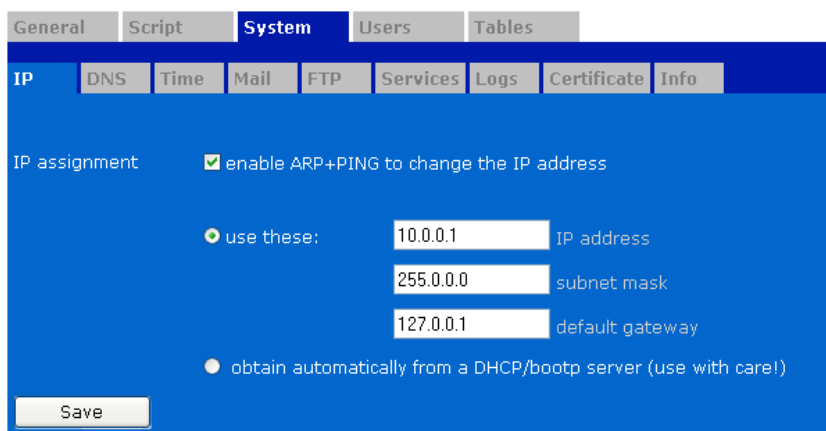
Two buttons are located under the edit box. Use "Save" to store the script with modifications or "Cancel" to discard any changes made.

Note that it might be more convenient to use a normal text editor while developing scripts. You can up- and download the files over FTP, see chapter 4.1.

## 3.4 System settings

This part of the configuration is split into a number of tabbed dialogs. They let you configure the system properties, ranging from the IP address to the system time and logging.

### 3.4.1 IP addressing



The screenshot shows the 'System' configuration page in FieldCommander. The 'IP' tab is selected, and the 'enable ARP+PING to change the IP address' checkbox is checked. Under the 'use these:' radio button, the IP address is set to 10.0.0.1, the subnet mask to 255.0.0.0, and the default gateway to 127.0.0.1. The 'obtain automatically from a DHCP/bootp server (use with care!)' radio button is unselected. A 'Save' button is located at the bottom left of the configuration area.

#### Terminology

IP: Internet Protocol  
DHCP: Dynamic Host Configuration Protocol  
Bootp: Bootstrap Protocol  
ARP: Address Resolution Protocol

These settings control the ethernet IP settings of FieldCommander. Be careful when making changes as you may lose access to your FieldCommander unit. Check the Installation Guide for detailed information on internet addressing and how to set it up in your network.

#### Enable ARP+PING

In situations where you don't have access to your FieldCommander unit or you have lost its address, ARP+PING provides a solution to change it from your workstation. When you uncheck the box, this feature is disabled which may lock you out of FieldCommander. Refer to the Installation Guide for detailed information on ARP+PING.

#### Manual IP settings

It is recommended to assign FieldCommander a fixed IP address. Choose this option and fill out the IP address, subnet mask and default gateway/router address. Set the default gateway to 127.0.0.1 when you are not using a gateway or router.

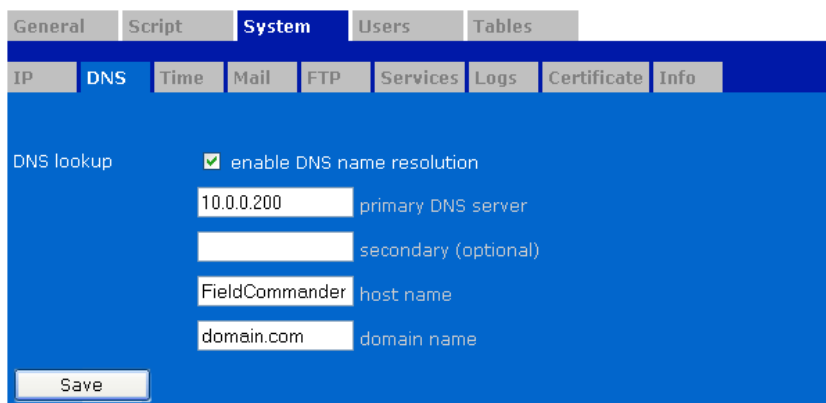
#### DHCP/bootp server

DHCP and bootp enable individual computers on an IP network to request their IP address from a server (the 'DHCP or Bootp server'). Consult your network administrator if you don't know if there is a DHCP or bootp server present.

**Caution:** Be aware that, when your FieldCommander is assigned a dynamic address, you may not know which address to enter in order to access it. Use this setting with caution.

You have to restart the system in order for the new settings to take effect. Note that you have to point your browser to the new address and log in again when you've changed IP settings.

### 3.4.2 DNS lookup



The screenshot shows the 'System' configuration page in FieldCommander, specifically the 'DNS' tab. The page has a blue background and a navigation bar at the top with tabs for 'General', 'Script', 'System', 'Users', and 'Tables'. Below the navigation bar, there are sub-tabs for 'IP', 'DNS', 'Time', 'Mail', 'FTP', 'Services', 'Logs', 'Certificate', and 'Info'. The 'DNS' sub-tab is active. The main content area contains the following settings:

- DNS lookup**:  enable DNS name resolution
- primary DNS server**:
- secondary (optional)**:
- host name**:
- domain name**:

A 'Save' button is located at the bottom left of the configuration area.

#### Terminology

DNS: Domain Name Service

URL: Universal Resource Locator

Domain Name Servers are queried to resolve the hostname to its equivalent IP Address. DNS is the basic "directory assistance" service of the Internet. DNS resolves domain names to the IP address (number) assigned to a specific machine on the Internet.

You'll need to enable DNS lookup if you want to use alphanumeric server and domain names (like ftp.yoursite.com) instead of numerical IP addresses (e.g. 10.0.0.1). This applies to references to IP addresses and URL's throughout FieldCommander.

#### Enable DNS lookup

To enable DNS lookup, check the box and fill in the primary and, optionally, the secondary DNS server IP addresses, host and domain name. Ask your network administrator for these details when you are in doubt.

#### Host name

The system name you want to pass through to the DNS server. Fill in the name you want to give to your FieldCommander, e.g. "FieldCommander".

#### Domain name

The domain name you want to pass through to the DNS server. When FieldCommander is attached to a LAN with its own domain name, fill in this domain name.

The current settings are saved by clicking the "Save" button.

### 3.4.3 Time settings

#### Terminology

NTP: Network Time Protocol

GMT: Greenwich Mean Time

UTC: Universal Coordinated Time

FieldCommander's internal clock is used in a variety of places in the system, to stamp acquired data, trigger events, show time in your web site, stamp entries in the log etcetera, so it is convenient to set it to the correct time. It is stored in UTC format and can be adjusted to your local time.

#### Time zone and daylight saving

Pick your local time zone here from the list, and check the box to enable daylight saving when it applies to your location. You have to restart the system in order for new time zone settings to take effect.

#### Manual system time/date

Adjust the internal clock here by manually entering the current time and date and clicking "Save". The time should be entered in 24 hour format: hh:mm:ss, the date as: mm/dd/yyyy.

#### Use time server to synchronize time

(in selected models in all models)

FieldCommander is capable of requesting the time and date from public time servers using the NTP protocol. Use this setting when you have access to a time server on your local network or when you have a permanent connection to the Internet. It ensures that FieldCommander always has the correct time.

When you have a time server on your LAN, it is advised to use this to synchronize FieldCommander. In other cases, visit <http://www.ntp.org> for a list of public NTP servers and choose one near your location for the shortest network path to minimize the time error. If you enable DNS resolving you can use a domain name here instead of the numerical IP address.

NTP servers communicate with one another using UDP with a destination port of 123. You'll have to allow UDP traffic on source/destination port 123 between your server and the time server with which you are synchronizing.

### 3.4.4 Mail server

The screenshot shows the 'Mail' configuration tab within the 'System' section. The interface has a blue background. At the top, there are tabs for 'General', 'Script', 'System' (selected), 'Users', and 'Tables'. Below these, there are sub-tabs for 'IP', 'DNS', 'Time', 'Mail' (selected), 'FTP', 'Services', 'Logs', 'Certificate', and 'Info'. The main area contains three input fields: 'SMTP server address' with the value 'mail.cer.com', 'port number (default is 25)' with the value '25', and 'return address (used as "from" field)' with the value 'fc3310@cer.com'. A 'Save' button is located at the bottom left.

#### Terminology

SMTP: Simple Mail Transfer Protocol

In order to give FieldCommander the ability to send e-mail messages, you have to configure the SMTP server settings here. Note that this tab is only available if the "enable outgoing mail" feature is set in the "Services" tab.

#### SMTP server

Specify the address of the SMTP server you want to use for outgoing mail. This can be an IP address or a domain name. In the latter case, you must have enabled DNS resolving.

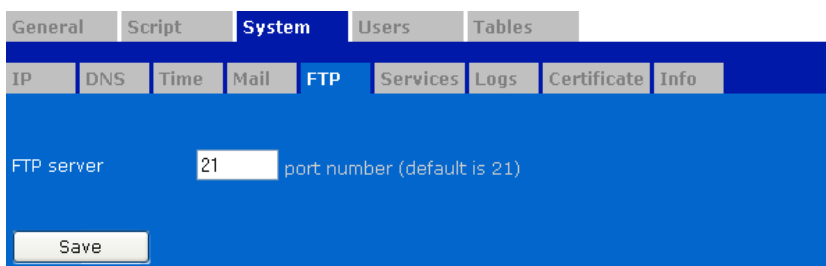
#### Port number

The default port for the SMTP protocol is 25. Some servers may use a different port, so you can change it here. When in doubt, leave it to 25.

#### Return address

Define a valid mail address here. It will be used in the "From" field of the messages FieldCommander sends. Replies to the messages will be directed to this address.

### 3.4.5 FTP server



#### Terminology

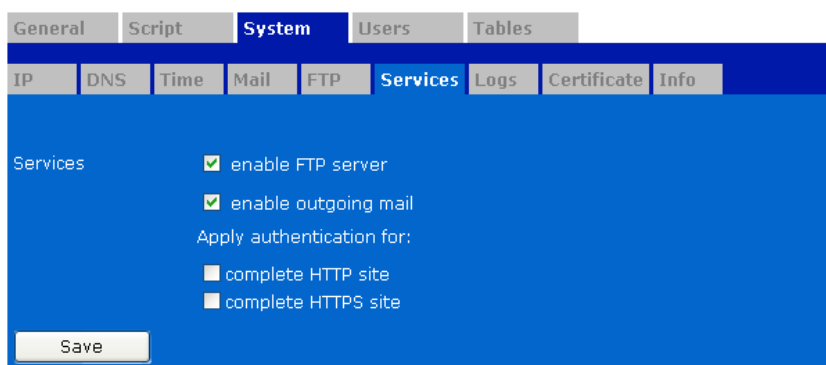
FTP: File Transfer Protocol

The built-in FTP server provides direct access to the disk of FieldCommander. You need it to upload your application specific scripts, web pages, images, log files etcetera. The FTP server is enabled by default, but it can be disabled for security reasons under the Services tab. Once disabled, the FTP configuration tab becomes inaccessible. Note that you can instead choose to limit the use of the FTP server to trusted users only. Refer to next paragraph for details.

#### Port number

The default port number to connect to FieldCommander's FTP server is 21, but you can change it if you need to. When in doubt, contact your network administrator or leave it to it's default value. Changes will take effect immediately after clicking the "Save" button.

### 3.4.6 Services



#### Terminology

HTTP: HyperText Transfer Protocol

From this dialog, you can enable FieldCommander's FTP, email and HTTP Authentication services. This tab is only available, when you log in with a user profile that is granted "admin mode" rights to the system settings.

#### Enable FTP server

Check to enable FieldCommander's built-in FTP server to upload your application specific scripts, web pages, images, log files etcetera. If disabled, the FTP configuration tab will be inaccessible to all users. More importantly, you cannot up- or download any files to or from your FieldCommander unit. Note that you can instead choose to limit the use of the FTP server to trusted users only. This can be configured in the user profiles. Refer to chapter 4 on File management for information on how to use it.

### Enable outgoing mail

FieldCommander supports sending email messages directly from a script. This function is disabled by default, but you can enable it by clicking the corresponding box. Note that if disabled, the email configuration tab will be inaccessible to all users.

### HTTP Authentication

Typically, web sites are open and accessible by anyone with a browser and a network connection to the server. The same applies to FieldCommander. In certain situations, however, you might want to control access to - certain parts of - its web server. This is achieved by basic HTTP authentication.

You can choose to require authentication for:

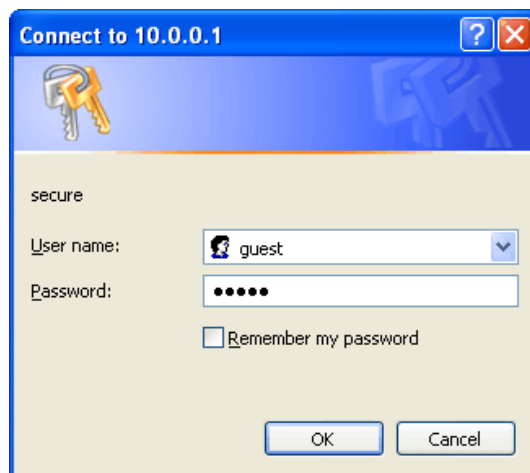
- *complete HTTP site*

When enabled, user authentication is required for all files in the "/www" tree. Disabled means authentication is only enforced for "/www/secure".

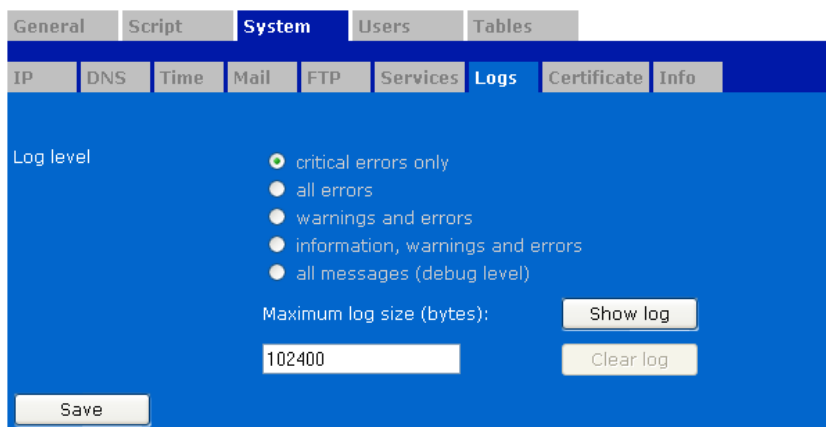
- *complete HTTPS site*

When enabled, user authentication is required for all files in the "/www-secure" tree. Disabled means authentication is only enforced for "/www-secure/secure".

In addition to these basic authentication settings, you can set access restrictions based on User Profiles. See paragraph 3.5 for details. When authentication is enabled and you request a file from the web server, your browser will prompt for a user name and password as shown in this dialog:



### 3.4.7 Logs



As FieldCommander cannot report anything to a screen, it can write errors, warnings and/or debug messages to a log file. You can view the log file from here and the script settings dialog. Note that increasing the level of detail, also increases the demand for system resources. You should only do it while you are testing or troubleshooting your application. This tab is only available, when you log in with a user profile that is granted "admin mode" rights to the system settings.

#### Log level

You can choose the level of information to include/store in the log. Entries are classified in five levels, in order of importance: critical errors, errors, warnings, information and debug messages. Messages by the script function *writeLog()* are of type "information". The default setting is "critical errors only".

Increasing the level of detail in the log, will decrease system performance and should only be done when you are testing or troubleshooting your application.

#### Log size

You can specify a maximum size of the log files to limit the consumed storage space. When the log exceeds the configured limit, it is "rotated". Check chapter 4.3 for more information on log file rotation. The log files are placed in the "/log" directory and can be accessed over FTP.

#### Show log

View the current content of the log file.

#### Clear log

Erase the log files. It can be useful to start with a empty log to easily spot the new entries.

Check chapter 4.3 for more information on the use of the log.

### 3.4.8 SSL certificate

General Script **System** Users Tables

IP DNS Time Mail FTP Services Logs **Certificate** Info

Credentials  use CER credentials  use custom credentials

Host name/address

Days valid  from today, 11/02/2005

Country code  international 2 character country code

State

Locality/city

Organisation

Department/unit  (optional)

FieldCommander is shipped with a so-called "self signed" SSL certificate by CER International used for HTTPS encryption. As the certificate may expire or the IP address of the device may change, you can generate a new certificate here.

#### Credentials

You can choose to create a new SSL certificate based on the default authority details from CER, or create a custom one by filling out the credentials yourself.

For "hostname", you should enter the IP address or fully qualified domain name (e.g. *fc3310.cer.com*) by which the **end user** (browser) accesses FieldCommander. The "country" should contain a 2-character international country code as defined in ISO 3166. The state, locality and organisation (all 64 characters maximum) are mandatory as well. The department field is optional and may be left blank.

#### Generate

When all information has been entered, the certificate can be generated. The new certificate replaces the currently installed one and will be used for all SSL connections after FieldCommander has been restarted.

### 3.4.9 System information

System details	
Model	FC33102030
Serial number	00 02 B6 19 72 B6
IP address	10.0.0.1
System time	11/02/2005, 15:25:22
Hardware platform	Platform 2
Software edition	Advanced
Software version	3.4.13
Resources	
Uptime	0 days, 0:10:39
Load average (5 min)	0%
Disk free	184632 kB
Disk used	37052 kB
Memory free	78076 kB
Memory used	45820 kB
Options	
Audio support	Yes
Camera support	No
MMS support	No
Modbus option	Yes

This page serves to provide detailed information of the system, available options and the current use of the system's resources.

## 3.5 User profiles

User profile	
<input type="text"/>	Add...
quest	Edit...
admin	Delete

The user profile settings let you set up and alter the user accounts and passwords. You can grant/prevent access to FieldCommander's resources.

#### Add a profile

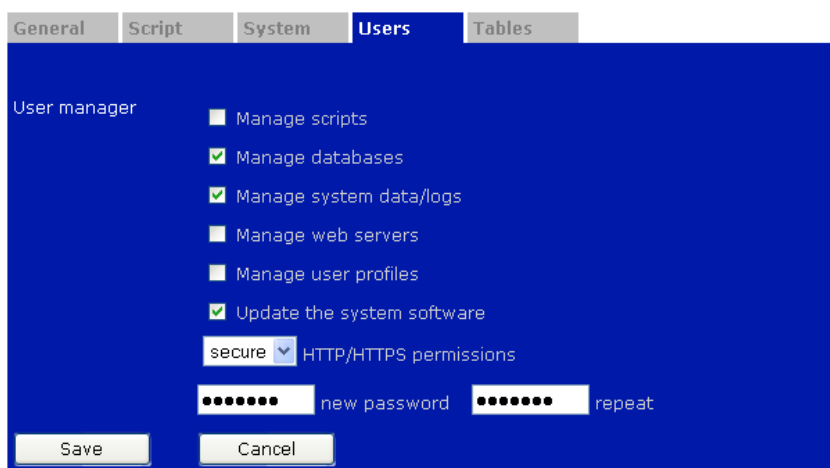
To add a new user, enter the new name (16 characters maximum) in the box and press the "Add" button. This creates an empty profile to which you can add permissions (see below). Note that the user names are case sensitive.

### Delete a profile

Select the name in the list click the "Delete" button. When you remove the profile of the current user, your session is terminated immediately.

### Edit a profile

Pick a user name from the list and press the "Edit" button to change the user's permissions. You cannot edit your own profile. Changes to a user profile becomes active the next time that user logs in. The picture below shows the Edit Profile dialog:



### System access

The options here relate directly to FTP access to various directories of the disk and indirectly affect the dialogs available in the System configuration. The table below lists the consequences of enabling the checkboxes.

option	FTP permissions <sup>*)</sup>	System configuration implications
<i>Manage scripts</i>	/script	Script tab
<i>Manage databases</i>	/database	Tables tab
<i>Manage system data/logs</i>	/log and /data	Services and Logs tabs under System tab
<i>Manage web servers</i>	/www and /www-secure	(none)
<i>Manage user profiles</i>	(none)	Users tab, user related functions in FCphp
<i>Update system software</i>	/update	Reboot button in General tab

\*) The FTP server itself can be switched off the System settings. In that case, nobody can log in.

### HTTP/HTTPS permissions

Access to the web servers (both normal HTTP and secure HTTPS) can be set in three modes:

- **none**: use is not allowed to access the web server when authentication is applied
- **normal**: allowed to authenticate for the web servers except the /secure folders
- **secure**: allowed to access files in the /www/secure and /www-secure/secure folders

As the table below shows, the access permissions are configured at two sides; the HTTP server authentication mode and user permissions. The web site authentication mode (configured in the System/Services dialog) defines on what level the security is applied:

view pages in web directories	apply authentication for web site	user HTTP/HTTPS permissions		
		none	normal	secure
/www/secure	n/a			☞
/www-secure/secure	n/a			☞
/www and below	HTTP checked		☞	☞
	HTTP unchecked			
/www-secure and below	HTTPS checked		☞	☞
	HTTPS unchecked			

☞ free access, no authentication required

☞ access granted after name/password authentication

☞ no access (authentication will deny access)

### Password

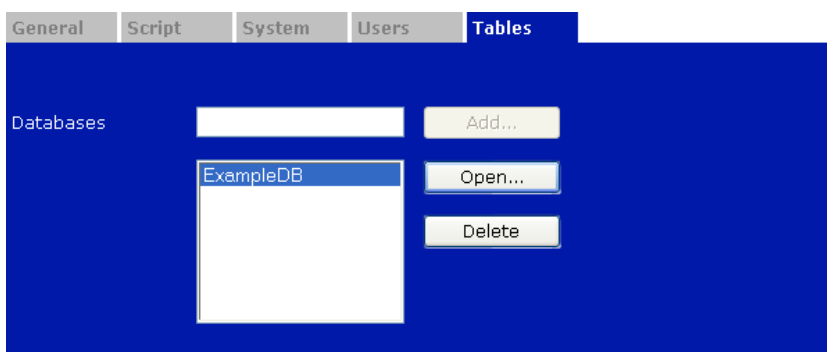
For new user profiles, you must pick a password and fill it out twice to reduce the risk of typing mistakes. In existing profiles, you can use the fields to change the password. Note that they are case sensitive.

All settings take effect immediately after pressing the "Save" button. To discard all altered settings and return to the User profiles index, click the "Cancel" button.

## 3.6 Database admin

From the Tables dialog, you can create, edit and delete databases and tables. Access rights to this dialog is defined in the User Profiles. When selecting the Tables tab, you will be prompted with the Databases dialog.

### 3.6.1 Databases



Here you can add, delete and open databases.

**Add**

Add a database by typing a unique name in the input field and clicking the "Add" button. Database names are case sensitive and must have a minimum length of 3 and a maximum length of 20 characters. Creating a database opens the Table dialog. You can store any number of databases on FieldCommander, as long as there is sufficient disk space available.

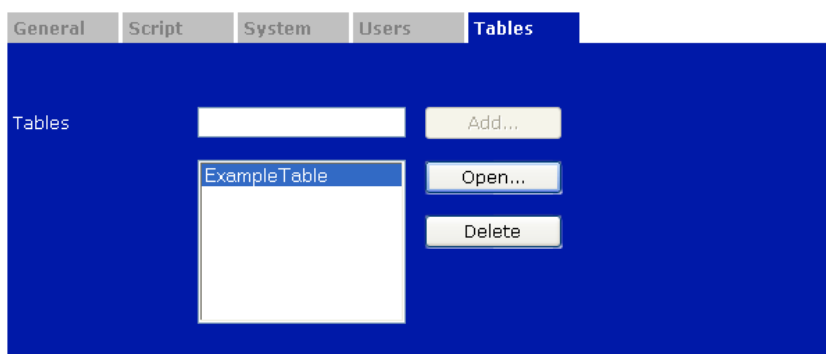
**Open**

To open an existing database, select it from the list and click the "Open" button. This will open the Tables dialog.

**Delete**

To delete an existing database, select it from the list and click the "Delete" button.

### 3.6.2 Tables



This dialog allows you to add, delete and open tables in the current database.

**Add**

Add a table to a database by typing a unique name in the input field and clicking the "Add" button. Table names are case sensitive and must have a minimum length of 3 and a maximum length of 20 characters. After adding a new table to a database, the Table properties dialog will open. Note that a single database can contain multiple tables.

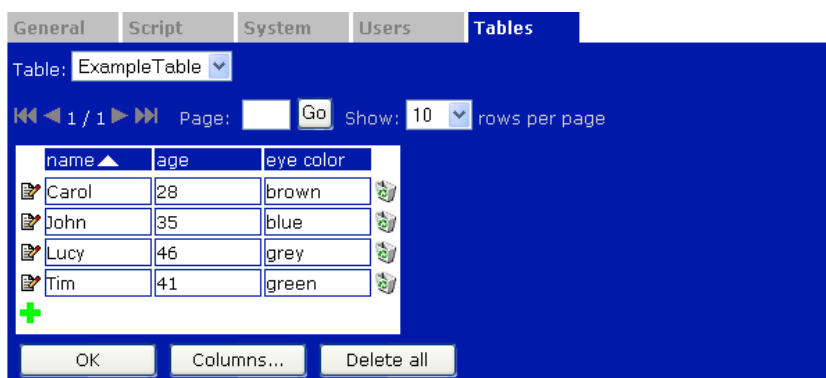
**Open**

Clicking this button will open the Table data dialog where you can view and edit data and table properties.

**Delete**

To delete a table and its contents, select it from the list and click the "Delete" button.

### 3.6.3 Table data editor



In this dialog, you can browse the contents of the table.

#### Navigating

For quick navigation between tables, you can pick a different table from the database using the drop down list in the top of the dialog. Longer tables are divided into several pages. You can choose the number of rows per page (10, 20, 50, 100) from the drop down list.

Using the and buttons you can browse through the pages. To jump to a specific page, enter the number in the box and press the "Go" button.

#### Sorting

You can click a column label to sort the table using this column. Click again to reserve the order. The sort column and direction is indicated by the symbols.

In addition, you can edit, add and delete rows:

#### Edit

Click the "Edit" button in front of the row you want to change. Now you can edit the content of the fields, except for the first field as it has a unique index value. If you want to edit this value, you will first have to delete the row, and add a new row with the correct value(s). All fields hold text strings with up to 200 characters. Save your changes by clicking the button or discard changes using the button.

#### Delete

To delete a complete row and its contents, click the "Delete" button next to the row you want to remove.

#### Add row

To add a new row to a database, click the "Add row" button. You will be taken to the Edit table data screen.

#### OK

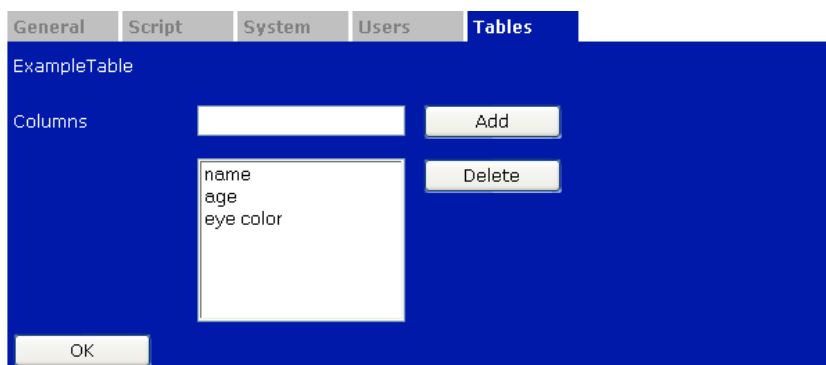
To return to the Tables dialog screen, click the "OK" button.

#### Columns...

Clicking the "Columns" button will take you to the dialog where you can add or remove columns of the table.

**Delete all**

Delete all rows in the current table. The table's structure - including the column names, number of columns and all table restrictions- remains intact.

**3.6.4 Edit table columns**

In this dialog, you can edit the columns of a table. This includes adding or deleting columns.

**Add**

To add a column to a table, type a unique name (20 characters maximum) in the input field and click the "Add" button. The first column is used as index for the table and will contain unique values. A table can have up to 32 columns.

**Delete**

To delete a table from the list, select it from the list and click the "Delete" button. The first column cannot be deleted as it is used as index. Note that upon deleting a column, you will also lose all data in that column.

**OK**

Click this button to return to the Table data dialog.

## 4. System administration

### 4.1 File management

FieldCommander is equipped with a spacious flash drive for the system software (referred to as firmware), logs and your custom application scripts, data and web site. This chapter explains how to access the disk and install your applications on it.

#### 4.1.1 Connecting with FTP

You can connect to the FTP server of FieldCommander using any common FTP client, or web browser which supports FTP. You need to provide the following details:

*User name* profile in user settings, "admin" and "guest" are preconfigured  
*Password* password defined for the user profile  
*IP address* address of FieldCommander, factory default is 10.0.0.1  
*TCP port* port number of FTP server, normally 21

To connect with a browser, type the following location in the address bar:

syntax: `ftp://username@ipaddress:port`  
example: `ftp://admin@10.0.0.1:21`

Now the browser will prompt you to supply the password, like this:



The password can be part of the URL but that is less secure as it might end up in your browsers history. As port 21 is assumed as default, you can omit the ":port" part of the line.

When the connection is made, the FTP client or browser gives a list of the files and directories on the FieldCommander's drive. Depending on the settings in the user profiles, you will see a listing similar to this:

data/	room for your application data
database/	room for databases
log/	logs will be are stored here
script/	room for your application scripts
update/	place to upload firmware updates
www/	room for your web pages, images etc. accessible by http
www-secure/	room for your web pages, images etc. accessible by https

You can create subdirectories in these directories. All file and directory names are case sensitive in FieldCommander.

Settings in the user profile you are connected with, define what you are allowed to see and do. The user may not be allowed to access, read and/or write certain directories and you may be locked out of FTP access completely. The FTP permissions are configured in the user settings of the web configuration. The built-in "admin" profile has all permissions.

#### 4.1.2 Installing your applications

Your custom application is typically build around a set of files which you will upload to FieldCommander disk. The files may be placed in separate directories, depending on their function.

##### Script files

FCscript files (extension .jse) are placed in the "/script" directory. These will appear in the Script settings dialog where you can start and stop executing them.

##### Web pages

FieldCommander can be accessed over either normal (http) and secure (https) connections. Requests over the normal (unencrypted) connection are served from the "/www" web root where secure requests (encrypted by SSL) are served from the "/www-secure" web root.

For custom web pages, you should choose to upload them to the normal or the secure web root. This include your html files, images, style sheets, Java applets, flash movies, report files, etcetera.

The "/www" and "/www-secure" folders have several fixed subdirectories:

admin/	place for your own configuration, redirection to _admin
applet/	(system) contains the applets
cgi-bin/	(system) supporting the configuration
secure/	files placed here are always protected by user authentication
tmp/	temporary files can be placed here (see also note below)

In " /www-secure" also:

_admin/	(system) configuration settings dialogs
---------	---

You can add subdirectories in the web root and in the secure directories. Note that all file and directory names are case sensitive in FieldCommander and spaces are not allowed.

There is a file called "index.html" installed in the web root (the welcome example page), which is shown by your browser when you do not specify a file name. You can replace it with your own home page. In case "index.html" does not exist, it tries to load "index.php", "index.htm" or "Default.htm" (in this order). If none of these are found, it will show a listing of the files in the directory.

When you request files in the "/secure" directory with a web browser, it will prompt you to authenticate with a user name and password. This password protection can also be set for the entire web root. Chapter 3.5 discusses the authentication system in detail.

### Supporting files

Files loaded or created/written by your script, are best placed in the "/data" directory. When you want files to be accessed through the web server, you must place them in the "/www" and/or "/www-secure" directories. Note that the file names are case sensitive in FieldCommander.

### Temporary files

FieldCommander is equipped with a flash drive. Writing files to this medium is not very fast and on every write, the flash drive 'wears out' a little. To speed up writing and to prolong the life of the drive, you can use special directories which are kept in volatile RAM memory instead of the flash drive. The directories are "/www/tmp" "/www-secure/tmp" and "/data/tmp" (1MB each). Note that files stored in these locations will be lost when the system is powered down or restarted.

## 4.2 Working with scripts

Scripts are ASCII text files written in your favorite editor on your preferred platform. While writing your application, you probably want to have the *FCscript Programmer's Guide* at hand as it is the book of reference for the language, objects and functions.

Upload your script (with extension .jse) to the "/script" directory so it will appear in the Script settings dialog where you can execute it. Just one script is active at a time, so you will have to stop a running script to be able to start another. To make smaller changes you can edit the script from the Script settings dialog.

**Note:** Changes to a running script do not take effect automatically. You'll have to stop the current script and restart it.

## 4.3 Log files

FieldCommander will report errors, warnings and debug messages though log files placed in the /log directory.

log/		
fcscript.log		FC script log with the current entries
fcscript_old.log		FC script backup log
php.log		PHP engine warnings and errors

### 4.3.1 FCscript logging

Entries from the FieldCommander script are classified in five escalated levels:

<b>critical</b>	system critical error, these should never occur; contact your local supplier when they are reported
<b>error</b>	error, most likely caused by an incorrect function call in the script; the application will most likely fail to run correctly
<b>warning</b>	non-critical error or suspicious situation, the application will run but possibly not as you expected
<b>information</b>	user messages written by the script function <i>writeLog()</i>
<b>debug</b>	detailed information, also reported during normal operation

You can choose the type of messages to store in the log from the System settings, in five levels. While testing or troubleshooting your application, you should set the log level to more detail, "information, warnings and errors" suits in most situations.

Increasing the detail level of messages to include in the log, will also increase the demand for system resources. It is advised not to set the detail level below "all errors" during normal operation. To limit the consumed storage space for the script log, you can configure the maximum size. When the current log file exceeds the configured size, it is copied to "fcscript\_old.log" and the original file is emptied.

### 4.3.2 PHP logging

Warning or error messages which occur while parsing PHP files are (for security reasons) not shown in the page but logged to the file "php.log". When you'd expect output from a PHP pages you are working on but a blank page appears, chances are that a parse error occurred.

## 4.4 Security

### 4.4.1 Passwords and profiles

To control access to the resources of FieldCommander, it uses user accounts also known as user profiles. A profile is protected by a user name and password, and stores access rights to various parts of the disk (over FTP) and web server. New users can be added and the permissions can be changed from the Users tab.

You must log in into the configuration pages with one of the user profiles. The password will be checked and, when successful, a PHP session is created with the logged in user's rights. Some of the configuration tabs in the dialogs can be hidden, depending on the user's rights.

#### 4.4.2 Secure web server

The standard HTTP protocol used to browser web pages using plain text communication, which may be monitored by eavesdroppers. HTTPS encrypts the data using either a version of the SSLv2/3 (Secure Socket Layer) protocol and the TLS (Transport Layer Security) protocol, thus ensuring reasonable protection from eavesdroppers, and man in the middle attacks.

FieldCommander runs both a standard (HTTP) and a secure (HTTPS/SSL) web server. You can decide for yourself if you want your application to be available over the standard, unencrypted connection, or over the secure connection. It is also possible to use a combination.

FieldCommander is shipped with a so-called "self signed" certificate by CER International. You can generate your own SSL certificate from the system configuration (see 3.4.8) or with a PHP call to `fcCreateCertificate()`.

#### 4.4.3 Firewall configuration

When you place FieldCommander behind a firewall, you might need to enable traffic to/from certain ports, depending on what you want to allow. The table below lists the ports used by FieldCommander.

port	protocol	direction (seen from FC)	usage
20+21 <sup>1)</sup>	TCP	inbound	FTP server
20+21	TCP	outbound	FTP client
25 <sup>1)</sup>	TCP	outbound	SMTP client
80	TCP	inbound	HTTP web server
123	UDP	inbound/outbound	NTP time synchronisation (in selected models)
443	TCP	inbound	HTTPS (SSL) web server
9743	TCP	inbound	WebNotify and WebGUI applet communication

<sup>1)</sup> configurable, port number in table is the factory default

Note that FieldCommander has its own firewall to prevent misuse or attacks to cause any harm.

## 4.5 Rebooting the system

There are several ways to reboot FieldCommander. First there is the "reset" button. Pressing the button will cause a hard reset of the system, similar to power cycling the unit. You will hear a beep after a few seconds. It can take around a minute before FieldCommander is up and running.

When you have no physical access to the unit, you can reset FieldCommander using the web based configuration. Open the system configuration and log with a user profile that has the

proper rights to reboot FieldCommander. The "Reboot system" button will now be shown at the lower right corner of the general screen. Note that your browser's connection with FieldCommander is interrupted for about a minute after clicking the reboot button.

In case a script was executing at the moment you resetted the system, it is interrupted which may cause your application to behave irregular. The script is automatically restarted when FieldCommander has finished booting. Note that you loose the application state and the content of the data buffers when rebooting.

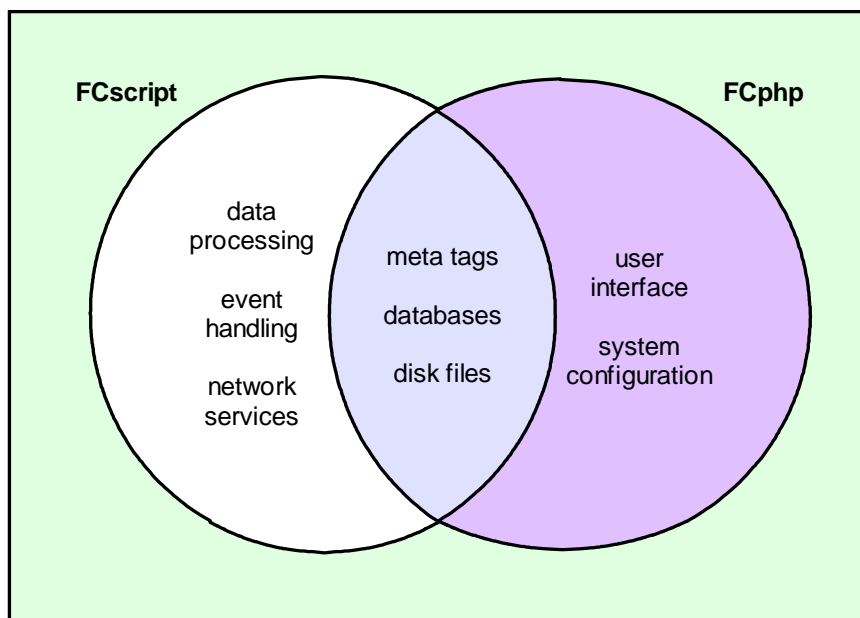
## 4.6 Updating the software

CER International releases software regular updates with enhancements, changes and fixes. These come in the form of binary images which can be uploaded directly to the unit. Log in to the FTP server and upload the image file to the "/update" directory (you'll need the appropriate user rights to do this). The new firmware is installed and takes effect after the system is rebooted.

Updates of the software and documentation are available from [www.cer.com](http://www.cer.com). Pay close attention to the guidelines and instructions which go along with the update, or you can cause damage to the system or may loose your valuable data.

## 5. Data sharing

FieldCommander features various ways to store and share data and information between the user interface (in FCphp) and underlying application (in FCscript).



The data read from the serial ports can be stored in data buffers (data processing). From there, FCscript can place the data or derived information, into a shared 'in memory' meta table, or into a relational database on disk. In case none of these meet your requirements, you can create your own data files and store them on the disk. FCphp, too, can read and write to the files, databases and meta tags.

This chapter discusses the features and you'll learn where to use which system.

### 5.1 Data buffers

The Buffer component is used to store and serve data units. Once data units are stored, you can browse and search the buffer for data in elements, and fetch it for further processing. Examples are searching for certain data strings, values, flags or time stamps. Typically, this is done in the `handleEvent()` routine of an FCscript.

#### 5.1.1 Accessing data

The Buffer component is used to store and serve data units. Once data units are stored, you can browse and search the buffer for data in elements, and fetch it for further processing. Examples are searching for certain data strings, values, flags or time stamps. Typically, this is done in the `handleEvent()` routine of a script.

The Buffer works with an internal pointer. With the `goto...()` and `find...()` functions the pointer can be moved. With `getBufferDataElement()` it is possible to fetch a data element from the data unit where the internal pointer is pointing to, `getBufferDataUnit()` retrieves a complete data unit. Below are a few examples of common situations.

Read all "data" elements from all data units, from the current position in the Buffer towards the end, and write them to the log. If the initial position in the buffer is undefined, the position is automatically set to the first data unit in the Buffer.

```
while(!gotoNext(bufferID)) {
    data = getBufferDataElement(bufferID, DT_RS, "data");
    writeLog("data:" + data);
}
```

Read all "data" elements from all data units, from the current position in the Buffer towards the start, and write them to the log. If the initial position in the buffer is undefined, the position is automatically set to the last data unit in the Buffer.

```
while(!gotoPrevious(bufferID)) {
    data = getBufferDataElement(bufferID, DT_RS, "data");
    writeLog("data:" + data);
}
```

Search from the start of the buffer for all occurrences of a data unit with the error flag set and write the value of the "timestamp" element and the "data" element to the log.

```
if(!gotoFirst(bufferID)) {
    while(!findFlags(bufferID, DT_RS, "control", DS_ERROR, 0)) {
        time = getBufferDataElement(bufferID, DT_RS, "timestamp");
        writeLog("Errors:" + time.sec + "." + time.usec + "data:" + data);
    }
}
```

Count the number of 'a' characters (decimal 96) in the buffer and write the result to the log.

```
count = 0;
if(!gotoFirst(bufferID)) {
    while(!findValue(bufferID, DT_RS, "data", 96)) {
        count ++;
    }
}
writeLog( count + " characters a in the buffer");
```

## 5.2 Meta tags

Meta tags are used to share information between your application script and user interface. As they are kept 'in memory', the tags cannot be used for permanent storage.

### 5.2.1 Using meta tags

The meta tags are listed in a table as key/content pairs. The meta tag names are the "keys", the value is the "content", in the form of a text string. You can read and write to this table using two straight-forward functions, *getMetaValue()* and *setMetaValue()*. The name tag names is limited to 64 characters, the content can be as long as 256 characters. The meta tags names are case sensitive, meaning that tag "MyValue" is different from "myvalue".

The meta tags can be read and written throughout FieldCommander in various ways:

FC technology	place	usage	see also
SSI	web page, htm/html	insert tag data (read)	chapter 7.1
FCphp	web page, php	read/write tag data	FCphp Guide
FCscript	application script	read/write tag data	FCscript Guide
WebNotify	web page, Java	pushed function call (read)	chapter 7.2
WebGUI	web page, Java	read/write tag data	chapter 7.3

### 5.2.2 System tags

There are several tags reserved by the system to provide information about the system itself. The tags start with an underscore and cannot be cleared or removed. Like all meta tags, the system tags are case sensitive.

meta tag name	description
_IPADDRESS	IP address of this FieldCommander, nnn.nnn.nnn.nnn
_HOSTNAME	hostname assigned to this FieldCommander
_TIME	local system time, HH:mm:ss (24 hour clock)
_DATE	current date, mm/dd/yyyy
_VERSION	version of the software (major.minor.patchlevel)
_SERIALNUMBER	serial number of this FieldCommander
_MODEL	model type of this FieldCommander
_LOAD	CPU load in percent, average of last minute
_LOAD5	CPU load in percent, average of last 5 minutes
_LOAD15	CPU load in percent, average of last 15 minutes
_UPTIME	system uptime, formatted as text
_UPTIMESEC	system uptime in seconds
_MEMFREE	available system memory in kilo bytes (kB)
_MEMUSED	allocated system memory in kilo bytes (kB)
_DISKFREE	available disk space in kilo bytes (kB)
_DISKUSED	allocated disk space in kilo bytes (kB)
_RS232_PORTS	number of RS232 ports in the system
_RS485_PORTS	number of RS485 ports in the system
_NDU<bufname>	number of data units in buffer <bufname> E.g.: "_NDUmybuffer" for buffer with name "mybuffer"

## 5.3 Databases

FieldCommander features a powerful database engine used to store structured information and share it between your application in FCscript and user interface based on FCphp.

### 5.3.1 Creating and editing

Databases, tables and rows of data are created and edited through the web interface under the "Tables" tab of the System configuration. Note that only users with access rights in the /database tab will see this tab. Chapter 3.6 explains it all. The FieldCommander PHP API, documented in the *FCphp Programmer's Guide* has all the functions to create your own database administration interface.

### 5.3.2 Permanent storage

Databases are stored in a file in the /database directory on FieldCommander's internal disk. You can store any number of databases, as long as there is sufficient disk space available. When you run out of disk space, you can also transfer your databases to a remote system for backup purposes, by using FieldCommander's FTP server. See paragraph 3.5 for details.

### 5.3.3 Accessing the tables

You can choose between the simplified functions for quick and easy access without needing to know anything about databases, and true SQL queries to do more complex things. The tables list both the FCscript and FCphp function names.

Easy access functions:

function	description
setConfigDatabase() fcSetConfigDatabase()	Set database to use by the following commands
getConfigTableRow() fcGetConfigTableRow()	Retrieve data from a specific row indicated by its index value and return an object/array containing the rows data
getFirstConfigTableRow() fcGetFirstConfigTableRow()	Retrieve data in the first row of a table and return an object/array containing the rows data
getNextConfigTableRow() fcGetNextConfigTaleRow()	Retrieve data in the next row of the result set and return an object/array containing the row's data
getLastConfigTableRow() fcGetLastConfigTableRow()	Retrieve data in the last row of a table and return an object/array containing the row's data
getPreviousConfigTableRow() fcGetPreviousConfigTableRow()	Retrieve data in the previous row in the result set and return an object/array containing the row's data
setConfigTableRow() fcSetConfigTableRow()	Add a row or change data in a row of a table
deleteConfigTableRow() fcDeleteConfigTableRow()	Delete a row from a table

SQL based functions:

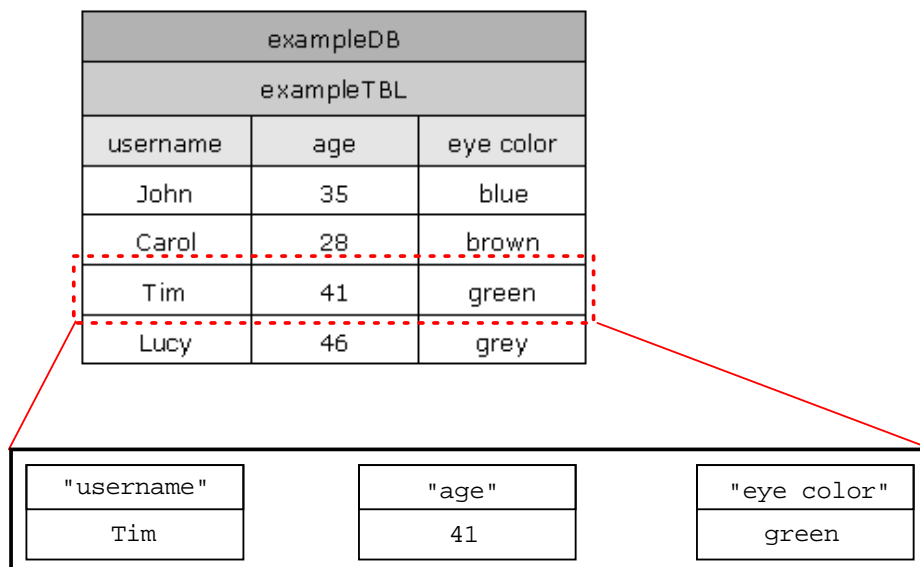
function	description
setConfigDatabase() fcSetConfigDatabase()	Set database to use by the following commands
configTableQuery() fcConfigTableQuery()	Execute a true SQL query. See Appendix D for syntax.
getConfigTableQueryRow() fcGetConfigTableQueryRow()	Get (next) row from the SQL query result set
configTableQueryFinalize() fcConfigTableQueryFinalize()	Free the current result set manually

With SQL queries you can do more advanced database actions such as selections with joins, create indexes, create autonumbered columns etc. Appendix D lists the supported SQL syntax.

The script commands are described in detail the *FCscript Programmer's Guide*, their PHP equivalents are documented in the *FCphp Programmer's Guide*.

### Using the returned rows

Several of the above functions return a row of data. In FCscript it is returned as an object, in FCphp as an associative array. As we'll see here, they are basically identical. Consider the example table below:



#### FCscript object:

row["username"]                      row["age"]                      row["eye color"]

#### FCphp array:

\$row["username"]                      \$row["age"]                      \$row["eye color"]

Database "exampleDB" contains a single table called "exampleTBL". Each column in a table uses a unique, case sensitive name; in this case there are 3 columns named "username", "age" and "eye color". There are 4 rows with data, with each row containing 3 separate entries called data fields. The values in the data fields of the first column are used to index the row they are in. Data is requested from a table one row at a time.

In this example, the first column "username" contains a list of names. To select and use data from this table, FieldCommander uses several script commands. In the following example, we will request the value "age" for "username" Tim.

First of all, you must tell FieldCommander which database to use. In FCscript the command would be:

```
setConfigDatabase("exampleDB");
```

You can then use getConfigTableRow() to retrieve information from a row and assign it to an object/array: in this case the object called *row* in Javascript or *\$row* in PHP. You must provide both table name and index value to select the proper row.

```
row = getConfigTableRow("exampleTBL", "Tim");
```

FieldCommander checks whether a row with that index value exists, and if so, returns an object/array with the fields of that row. In the example, the object *row* and array *\$row* contain the data from row with index value "Tim".

### Limits and restrictions

The following restrictions apply:

- up to 20 databases total
- up to 20 tables per database
- up to 32 columns per table or result set
- up to 1000 rows per table or result set
- up to 200 characters in a value/field

The database, table and column names are limited to 32 characters. Special characters (like quotes and double quotes) are not allowed. White spaces are allowed but make sure to use double quotes around the table and column names when passing them in a SQL query, eg:

```
SELECT * FROM "table name";
```

The values in a row are limited to 200 characters. Special characters are supported but special care should be taken with the single quote character as it needs to be escaped, see below.

### Escaping of quotes

The single quote character (') should be escaped by adding another single quote in order not to confuse the database engine. With `getConfigTableRow()` and `fcSetConfigTableRow()` it's done automatically but when you use SQL queries, you'll have to make sure to pass a query which is properly escaped. The escaping can be done by a single call with a regular expression, like this:

In FCscript:

```
str = "What's up!";  
str = str.replace(/'/, "'");
```

In PHP:

```
$str = "What's up!";  
$str = preg_replace("/'/", "'", $str);
```

## 5.4 Data exchange

The buffer, meta tags and database all share data *inside* of FieldCommander. There are various ways to share the information with external devices and systems, by e-mail, file transfer, text messaging etc. These features are subject in chapter 6.3.

## 6. Application script features

### 6.1 Data processing

#### 6.1.1 The flow of information

FieldCommander streamlines the flow of your information. All incoming data is packaged in so called *data units* and carried along the configured components. The components allow you to convert the data into information, filter it, and trigger events on specific situations. The information can be stored to a buffer which can be retrieved in event handling.

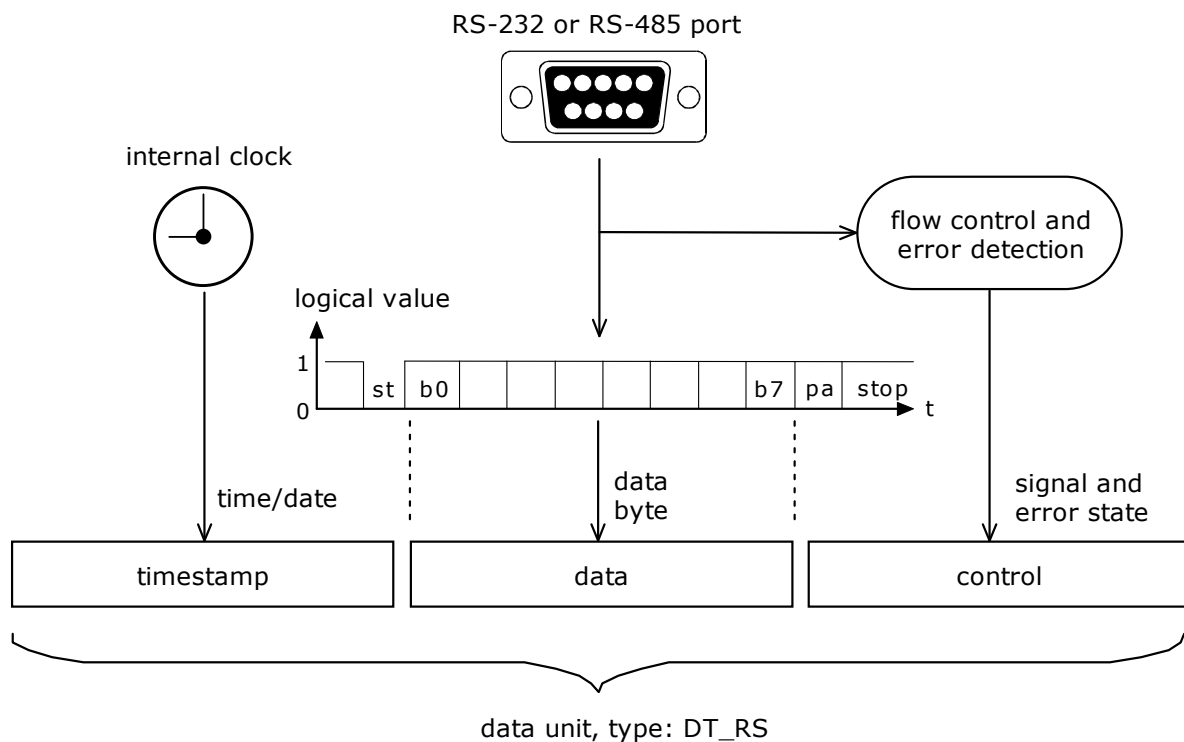
This chapter introduces you to the terminology and is concluded with an example network which puts them into place.

#### Data units

The information in FieldCommander is carried by data units. They originate from physical hardware, the RS-232 and RS-485 ports, and form streams flowing through a network of installed components.

Every data frame in RS-232 and RS-485 asynchronous communication, consists of a number of data bits, start- and stop bits to control synchronization and a parity bit. Next to the data lines, they define a set of lines to control the communication, called the *control signals*.

A data unit in FieldCommander consists of fields called *elements*. The RS-232 and RS-485 ports generate data units of type DT\_RS which contain the three elements: data, control and time stamp, as shown in the picture below.



## Data types

The RS-232 and RS-485 ports use the same data type (DT\_RS), though the definition of flags in the control element differs. In the appendixes you'll find the details on the two interfaces.

FieldCommander can handle different data types. The incoming serial data is of type DT\_RS, but there is a way to convert them: converter components. The ASCII frame converter (see paragraph 6.1) translates DT\_RS data units in a new data type called DT\_AF, containing a time stamp and a data string. When an external modem is used, text messages can be received. They are stored in datatype DT\_SMS.

FieldCommander supports the following data types:

data type	element name	element type
DT_RS	timestamp	Time
	data	Value
	control	Flags
DT_AF	timestamp	Time
	data	String
DT_SMS	timestamp	Time
	sender	String
	data	String

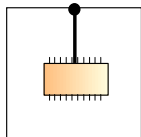
Optional protocol converters typically add one or more data types. Background on the data and element types is found in the *FCscript Programmer's Guide* and the appropriate *Option Guides*.

### 6.1.2 Components

FieldCommander's data processing is build around 'blocks' or 'components'. A set of components form a network through which a stream of data units flow. You can link the components in any way you want to, providing extreme flexibility. A maximum of 16 (for FC-SWS) or 64 (FC-SWA and FC-SWE) building blocks can be used in a network for each port, plus a total of 100 timers.

In order to understand the construction of such a network, you will need to know more about the functional characteristics of the separate blocks that make up a network. The components are introduced in the following part of this chapter.

#### Local port



**Name:** LocalPort  
**Purpose:** interface driver for physical port, the source of data units  
**Input:** serial data  
**Output:** stream of data units

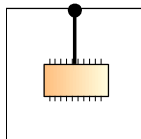
The Localport component provides the interface between the physical serial ports and the streams. Serial data frames enter the port where they are converted into a usable stream of data units, which can be processed by the next component in the network.

Apart from receiving data, the port can also be used to send data to serial devices. You can send (strings of) characters to a defined port number. The interface driver translates that information

into raw serial data and transmits it. By default the data will be acquired in 'raw' form, data type DT\_RS, for both the RS-232 and RS-485 ports.

Port 0, used in the example script where it simulates an external device, is reserved for demonstration purposes and should not be used in your custom applications. Other ports and their properties vary among FieldCommander hardware platforms. Refer to the appropriate Installation Guide for details.

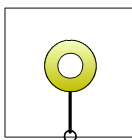
### Modem port



**Name:** ModemPort  
**Purpose:** interface driver for modem port and source of SMS data units  
**Input:** serial data  
**Output:** SMS data units

The ModemPort is a special type of LocalPort. When a modem is connected to the RS-232 port the data contains text messages of the DT\_SMS type.

### Buffer

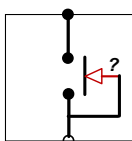


**Name:** Buffer  
**Purpose:** store and serve data units  
**Input:** stream of data units  
**Output:** none

The Buffer component is used to store and serve data units. Once data units are stored, you can browse and search the buffer for data in elements, and fetch it for further processing. Examples are searching for certain data strings, values, flags or time stamps. Typically, this is done in the *handleEvent()* routine of a script.

You can configure the Buffer's capacity and wrapping mode. In *linear mode*, data is added to the buffer until it is full. Once the buffer has reached the maximum capacity, it triggers an event so you can take responsive action in the script. In *ring mode*, the buffer is never full. It just overwrites the oldest data units, replacing it with the most recent data units in the stream.

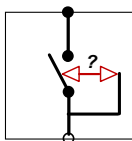
### Filter



**Name:** Filter  
**Purpose:** select data unit, stop unwanted information  
**Input:** stream of data units  
**Output:** selected stream of data units

The Filter component is used to select single data units. Only data units matching the (combination of) condition(s), are passed to the next components in the network. The other data units are dropped. Conditions can be defined on any combination of values, string matches, times or bit states. Refer to Conditions and Expressions (2.3) for details on conditions.

### Gate



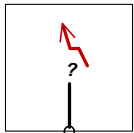
**Name:** Gate  
**Purpose:** select data units, stop blocks of unwanted information  
**Input:** stream of data units  
**Output:** selected stream of data units

The Gate component can be used to select blocks of data units in a stream to pass on to the next block in the network. It uses start and stop conditions to open and close the gate, respectively.

A closed gate drops all data units until a matching expression is detected to open the gate. When the gate opens, the data units are passed to the next component(s) in the network, until the expression to close the gate is matched. The data units causing the gate to open and close are also passed.

Conditions can be defined on any combination of values, string matches, times or bit states. Refer to Conditions and Expressions (2.3) for details on conditions.

**Data trigger**

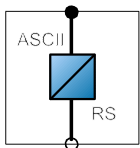


**Name:** DataTrigger  
**Purpose:** generate an event on condition match  
**Input:** stream of data units  
**Output:** none

The Datatrigger is used to generate an event when one or more elements in a data unit match the specified conditions. The generated event is handled by the event manager so you can take responsive action in the script. Typically, this is done in the *handleEvent()* routine.

Conditions can be defined on any combination of values, string matches, times or bit states. Refer to Conditions and Expressions (2.3) for details on conditions, and the chapter on Event handling on how to set up the script.

**ASCII frame converter**

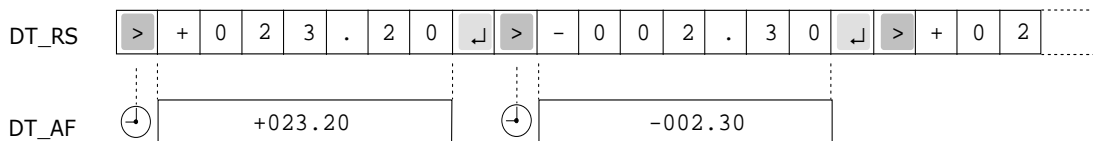


**Name:** Converter, type: ASCII\_FRAME  
**Purpose:** convert single bytes into an ASCII string  
**Input:** stream of data units of type DT\_RS  
**Output:** stream of data units of type DT\_AF

In the RS data type, every byte is represented by a data unit. Most communication protocols use single bytes or strings to indicate the start and end of a data frame. The ASCII frame converter, is able to convert the stream of data units with single bytes, into data units with a complete string frame. You can pass the start & end strings and/or the frame length.

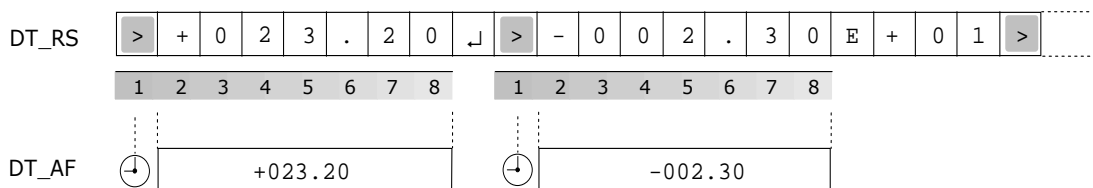
Example 1: start/end flags

start >  
 end ↓  
 length --



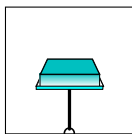
Example 2: fixed frame length

start >  
 end --  
 length 8



The resulting ASCII frame data unit has a data element that contains a string of characters (length max 256) and a time stamp of the first DT\_RS character of the frame start. The information of the control element is not part of the DT\_AF data. You can configure the converter to either include or exclude the start and/or end flags in the ASCII frame.

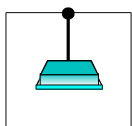
### Export



**Name:** Export  
**Purpose:** share information with other FieldCommander devices  
**Input:** stream of data units  
**Output:** stream of data units over TCP/IP

FieldCommander devices can be connected to form a distributed network. The Export component will set up a TCP/IP connection and export all data units in the stream. These data units are imported by another FieldCommander unit using a Remote port component.

### Remote port (in selected models)

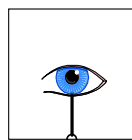


**Name:** Remote port  
**Purpose:** import data streams from other FieldCommander devices  
**Input:** stream of data units over TCP/IP  
**Output:** stream of data units

FieldCommander supports importing data streams from other FieldCommander devices. Streams of data units exported by another FieldCommander unit using an Export component, can be imported as a normal stream of data units via the Remote port component.

Like a Local port, a Remote port component always starts a new network of components. It cannot be added as an extra data stream in an existing network of components. Up to 12 remote channels can be used in a single script, with a maximum of 64 components in a network.

### Viewpoint (in selected models)



**Name:** Viewpoint  
**Purpose:** export data from the data stream to the WebGUI  
**Input:** data units  
**Output:** data elements over a TCP/IP connection to the WebGUI

The Viewpoint component is a coupling point for exporting data from FieldCommander's data stream to the web based graphical user interface, referred to as WebGUI. A Viewpoint is placed in the stream like most other components.

A Viewpoint enables you to visualize the data in the stream in real-time. Data units are received by the Viewpoint and the contents of the different elements are streamed directly to the WebGUI, which in turn presents them to the screen.

When the WebGUI applet is started, it registers itself to all its sources, including viewpoints. When a Viewpoint in a stream receives a new data unit, it is sent to all applets that registered themselves to the Viewpoint. For more information on the WebGUI, check chapter 7.3.

### 6.1.3 Conditions and expressions

Throughout the introduction to data units and components, the terms "condition" and "expression" have been used several times. They are used to control the filter, gate and trigger components and require additional explanation.

#### Conditions

A condition specifies a data state or occurrence which results in a logical (Boolean) "true" or "false" value. You can specify the following conditions on the elements of a data unit:

condition	function	options	type
value	compare element value with a given value	equal, greater, greater or equal, less, less or equal	long
time	compare element time stamp with a given time	relaxing time window	timeval
flag	compare element value with given bit masks	set bit mask, unset bit mask	flags
string	compare element text string against a given text	full string, sub string case sensitive, ignore case	string

#### Expressions

An expression is a logical combination of one or more conditions using Boolean syntax. They combine conditions in a single command by the use of following logical operands:

operand	function
	logical OR
&&	logical AND
!	logical NOT

In addition, you can use brackets "()" to change the order of evaluation of the operands.

#### Examples

- "condA"  
The above expression evaluates only one condition; it is true when condA is true
- "( condA || condB ) && !condC"  
This expression is true when **either** condA **or** condB is true but **only** when condC is false (not true).
- ""  
An empty expression is always true; this can be useful in combination with a DataTrigger as it will generate an event on every data unit that passes.

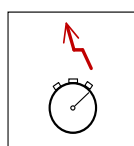
## 6.2 Event handling

An important concept of FieldCommander is that it's event driven. Events are situations detected ('triggered') during data processing or by timers. All events are handled by the Event manager which relies on your script to process the situation according to your needs.

### 6.2.1 Timer events

In the previous chapter you are introduced to the components which process data. Apart from being triggered in the data streams, events can also be generated by the timer clock of FieldCommander. A total of 100 timers can be used in your script which can be started as either an interval timer, or a clock timer.

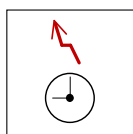
#### Interval timer



**Name:** IntervalTimer  
**Purpose:** generate event on timer interval  
**Input:** none  
**Output:** none

The interval timer is used to trigger events at a given interval. The interval can be as short as 10 milliseconds. This timer is often used in combination with sending data in order to poll sensors or devices.

#### Clock timer



**Name:** ClockTimer  
**Purpose:** generate event on time of day  
**Input:** none  
**Output:** none

With a clock timer enables you can generate an event on a wall-clock time. This works independent from the time stamps in the data stream. When you want to trap incoming data on a certain time of day, you can use a Data Trigger with a time condition.

### 6.2.2 Event callbacks

Typically, your script will include a central callback function named *handleEvent()*. This function is called every time an event occurs and indicates the source and type of the event. The *FCscript Programmer's Guide* explains how to use the event callback function in detail.

The following table summarizes the available event sources and conditions that trigger them.

event source	generated when...
Buffer (LINEAR mode)	the buffer is full, so no new data units can be stored until it is cleared
DataTrigger	a data unit matches the defined condition(s)
Trigger	a certain time or time interval elapsed
WebGUI	a hotrect is clicked by an end user
PHP page	fcTrigger() is called from a PHP page
Script	trigger() is called in the script itself

event source	generated when...
Modem	various causes, when connected, ring, disconnected etc
Email	reports success or failure after sending email message

## 6.3 Features

In the previous parts of the chapter you learned how FieldCommander acquires data and manages events. This part explains all functions and services which are available to create a powerful application.

### 6.3.1 Data from streams

The data from the local ports is captured in buffers to be used in FCscript.

#### Access data

In order to actually access and use data units of a stream in the script, its through a buffer component. A buffer has several services you can use in the event handling of your scripts:

- browse: goto first, last, next or previous data unit in buffer
- search: match data element's string, value, flag or time in buffer
- read: return content of element of current data unit

See paragraph 5.1 for more on the data buffers.

#### Transmit data serially

In many applications, you will want to send commands (requests) to the attached serial devices. The *sendString()* function allows you to send patterns (strings) of data to the local ports. Typically, this is done periodically in order to poll the status of a device or sensor. Most of FieldCommander Software Option include command to send information to the devices, dedicatd to the specific protocol.

### 6.3.2 Data exchange over network

The ethernet (TCP/IP) interface of FieldCommander serves as a vehicle for a number of different ways to exchange the data in FCscript with external systems:

#### Send e-mail

FieldCommander allows you to send e-mail messages with include HTML formatting, multiple recipients and attachments using a few straight forward function calls in FCscript. You pass the subject, file names and address(es) of the recipient(s). The body texts are predefined messages texts which can be uploaded via FTP or generated from the script itself.

You'll need to enable and set up the ability to send mail in the system configuration first. You must provide the target SMTP server, e-mail address of the sender, proper gateway address and DNS server when applicable.

#### Transmit data over TCP

Just like the *sendString()* function works to send texts over serial lines, *tcpSendString()* will transmit a text string over an exisiting TCP connection. The plain TCP socket connection is

established by calling *tcpConnect()* first. At the other side of the connection you'll need a simple TCP server socket to receive the text strings.

### Send and retrieve files (in selected models)

FieldCommander features an FTP client which enables you to send and receive files from the script. The functions *putUrl()* and *getUrl()* allow you to download and upload files from any FTP or HTTP server. This can be used to store data files on remote servers, or get files from a remote FTP/HTTP server. You can upload log files to a central FTP server to save space on FieldCommander's disk, or download data files to be used in your application.

### Export in XML (in selected models)

The Extensible Markup Language (XML) is the universal format for structured documents and data on the Web. XML is a set of rules (you may also think of them as guidelines or conventions) for designing text formats that let you structure your data. Like HTML, XML uses *tags* (words bracketed by '<' and '>') and *attributes* (of the form *name="value"*). While HTML specifies what each tag and attribute means, and often how the text between them will look in a browser, XML uses the tags only to delimit pieces of data, and leaves the interpretation of the data completely to the application that reads it.

FieldCommander can send your structured XML documents over a TCP socket stream to a 3<sup>rd</sup> party device or application which reads it. Analogue to the web pages, FieldCommander will parse XML templates and replace the meta-tags with up-to-date values. It uses the same meta-table as discussed in chapter 5.2.

Your XML template, uploaded to the */DATA* directory, may look like this:

```
<XML>
  <logentry>
    <room><!--#meta R20ROOMNAME --></room>
      <temperature><!--#meta R20TEMPERATURE --></temperature>
    <humidity><!--#meta R20HUMIDITY --></humidity>
  </logentry>
  <logentry>
    <room><!--#meta R21ROOMNAME --></room>
    <temperature><!--#meta R21TEMPERATURE --></temperature>
    <humidity><!--#meta R21HUMIDITY --></humidity>
  </logentry>
</XML>
```

The structure of your XML template totally depends on what the receiver of the XML documents expects. You can upload your XML template to the data directory with an FTP client of your choice.

The actual transfer of the XML document is started with the script function *exportXML()*. Before exporting the XML document, all meta-tags are replaced with the current values as they are listed in the meta-table.

### Exporting data units

FieldCommander systems can send information in data units over TCP/IP networks by way of the "export" and "import" functions. This way you can build a distributed network of FieldCommanders which share information.

The client (producer) can use *addExport()* to automatically establish a connection with the server

(consumer) and export all data units in the stream. Alternatively, you can *tcpExport()* in combination with *tcpConnect()* in order to send single data units.

FieldCommander to FieldCommander linking is a point-to-point type of connection. A single Export port connects to a single Remote port. A single FieldCommander unit however, can handle several Export as well as Remote port components at once in a single script.

#### **Importing data units** (in selected models)

The receiving FieldCommander using a Remote port acts as a server. The data units coming from a Remote port component form the starting point of a new network of components, which can be treated in the same way as any other data stream in FieldCommander. You can add filters, gates, triggers, buffers, converters and viewpoints to the stream.

### **6.3.3 Telephony**

Although FieldCommander has no internal telephony hardware, it can work with modems hooked up to one of its RS232 ports.

#### **Call setup**

You can set up voice calls and accept incoming calls. In combination with the audio features (see next paragraph) this allows you to add IVR (interactive voice response) capabilities to your application. When the network and modem supports it, you can retrieve the Caller ID of the incoming call.

#### **Send and receive text messages**

When you hook up the (optional) modem for cellular communications, incoming SMS text messages are automatically retrieved from a data stream in FieldCommander. The same modem can be used to send out a text message with a single call to the script function *sendSMS()*. The modem should support the ASCII based AT command set, defined in the ETSI GSM 07.05 protocol.

Even without a modem, you could send SMS messages. There are numerous organizations on the Internet who provide services to forward (subjects of) e-mail messages to cellular phones as an SMS text message.

#### **Send multimedia messages** (in selected models)

The GSM modem (optional), cannot only send short text messages but also pictures. This is called multimedia messaging or MMS. With the *sendMMS()* function you can pass a picture along with a text message and subject line to one or more recipients. A recipient can be either the number of a mobile phone or an e-mail address.

### **6.3.4 Audio**

(in selected models)

FieldCommander can use the built-in audio interface (available on selective models) to play audio and detect DTMF tones. With this, you can create interactive voice response applications.

The audio clips should be encoded as follows:

file format	WAV
sample rate	22050 Hz
sample size	16 bit
channels	mono
encoding	PCM encoding

DTMF tones, the audio signals generated when you press a button of a touch-tone telephone, can be recognized and passed as the character value.

### 6.3.5 Imaging

By hooking up an USB camera (optional), you can record a picture as easy as with a conventional photo camera. A simple call to *getImage()* is enough to take a picture and write it to the given file. Optionally the picture is overlaid with a text string, which you can use as time/date stamp, for example. The image is stored in JPEG format with a resolution of up to 640x480 pixels. A total of four camera's can be connected at the same time.

You can even merge the picture with a transparent overlay image using *overlayImage()*. Use this to add a logo or a decorative frame.

## 7. User interface features

### 7.1 Dynamic web pages

The built-in web server(s) not only host the system configuration of FiledCommander (subject in chapter 3) but can also serve a custom user interface. Your own interface can be composed of plain HTML pages, dynamic pages with PHP and even with real-time graphics using the WebGUI applet.

#### 7.1.1 SSI meta tags

Meta tags are used to share simple data between script and web pages. You can fill the tags from the FCscript and include them as dynamic content on your web page. The meta tags are listed in a table as key/content pairs. The meta tag names are the "keys", the value is the "content", in the form of a text string. You can read and write to this table using FCscript functions `getMetaValue()` and `setMetaValue()`. All meta tags names are case sensitive.

You can add meta tags in your HTML pages by using the SSI directive "#meta":

```
<!--#meta TANK_A -->
```

Here, `TANK_A` is a tag name. Note that meta tags are case sensitive. When a meta tag does not exist, it returns an empty string.

An alternative way to add dynamic content to your site, is by inserting a complete file into the web page. You can create text files in the script and insert them in a HTML page using the "#include" directive:

```
<!--#include file="errorlog.txt" -->
```

For example, if you have an application that keeps track of error messages, and the script writes the message to a file named "errorlog.txt", you can include the file contents into your HTML page as follows:

```
<html><body>
<b>The error message log file:</b>
<pre>
<!--#include file="errorlog.txt" -->
</pre>
</body></html>
```

The path points to the web root. You can prefix the file name with a path but it cannot start with a slash (/). Note that file and directory names are case sensitive.

#### 7.1.2 PHP engine

To enhance the site even more, the web server supports the PHP language. Use PHP to create full featured dynamic pages which can include information from the FieldCommander's database include information to your pages, and forms to set new values to meta tags.

#### PHP extensions

FieldCommander uses PHP version 5.0 and contains the complete core function library and several extentions:

- image functions (based on GD library)
- session handling functions
- XML parser functions
- Zlib compression functions

We refer to FieldCommander's hypertext language as **FCphp** to set it apart from the common PHP. FCphp is not limited to the core PHP functions alone, it adds an API of about 100 functions unique to FieldCommander. Most importantly, they provide access to the database and configuration settings of FieldCommander. This guide documents these functions in detail.

### Templates

The web based system configuration of FieldCommander is built with FCphp and is a good example of what you can do with PHP. The source of these pages is available through the FTP interface (/www-secure/\_admin) so you can use them as a template to build your own user interface or serve.

FieldCommander's PHP functions are documented in the *FCphp Programmer's Guide*. For information of the core PHP language we advice to check out [www.php.net](http://www.php.net) or one of the many good books on this subject.

## 7.2 Real-time web page updates

(in selected models)

Web pages are loaded - and filled with up to date information - at the moment they are requested by your browser. When new events occur or information in the databases is updated, the page in the browser stays put, unless you reload the page manually.

### Update only when necessary

It would be possible to force the browser to reload the page at a given interval but then you will request all data, even when nothing has changed which creates an extra load for your browser, network and the webserver.

With the unique WebNotify feature of FieldCommander, you can create a web page which will be informed of changes without the need to 'poll' or refresh constantly. Just have to set at meta tag in the FCscript, and WebNotify will call a (Javascript) function in your web page.

### WebGUI in a web page

The WebNotify applet is embedded into a web page and loaded from FieldCommander. The page file should the following code::

```
<APPLET WIDTH="0" HEIGHT="0" MAYSCRIPT="true">
  <PARAM NAME=CODE VALUE="WebNotify.class">
  <PARAM NAME=ARCHIVE VALUE="/applet/WebNotify.jar">
  <PARAM NAME=TYPE VALUE="application/x-java-applet;version=1.4">
  <PARAM NAME=IPADDRESS VALUE="<? echo fcGetMetaValue('_IPADDRESS'); ?>">
  <PARAM NAME=XMLCONFIG VALUE="notify.xml">
</APPLET>
```

This loads and initialises the WebNotify applet. Parameter IPADDRESS refers to the FieldCommander where to get notifications from. XMLCONFIG refers to the file **notify.xml** which has the applet settings.

## XML configuration

WebNotify is configured and created with a XML based configuration file. As the XML configuration file is merely a structured text-based document, you can create it using your favourite text editor and upload it to FieldCommander using FTP. The standard syntax for the XML configuration file looks like this:

notify.xml:

```
<?xml version="1.0"?>
<fc_applet_definition>
<variable type="metatag">
    <source_name>measured</source_name>
</variable>
</fc_applet_definition>
```

The code above instructs the WebNotify applet to watch for changes in the meta tag **measured**. It will call the javascript function with the same name.

## Calling Javascript

In this example, notify.xml (above) defines that new values of a metatag named "measured" will be passed to the applet. The applet calls the javascript function with this name. The value of the metatag is passed as the argument of this function. The function called **measured()** defined here forces a reload/refresh of the main frame.

```
<SCRIPT LANGUAGE="javascript">

    function measured(message) {
        parent.mainframe.location="main.php";
    }

    function WebNotify(message) {
        alert(message);
    }

</SCRIPT>
```

Besides the metatags/functions defined in the XML configuration, the applet may call the function WebNotify() to pass (error) messages. You can handle them the way you like but it is advised to at least define this function to avoid javascript errors when it is missing.

## WebNotify requirements

The WebNotify applet requires the Java 2 Runtime Environment (JRE) to work. It is developed by Sun Microsystems and shipped with FieldCommander on the documentation CD, or freely available from [java.sun.com](http://java.sun.com). The applet connects to FieldCommander on TCP port 9734 and exchange data (see also paragraph 4.4).

## 7.3 Graphical user interface

(in selected models)

FieldCommander's web server hosts custom web pages to visualize selected data on the screen. Dynamic data in web pages is linked by meta tags to a meta table in FieldCommander, which is updated in real-time. The values on the screen are refreshed by the web browser upon reloading the web page.

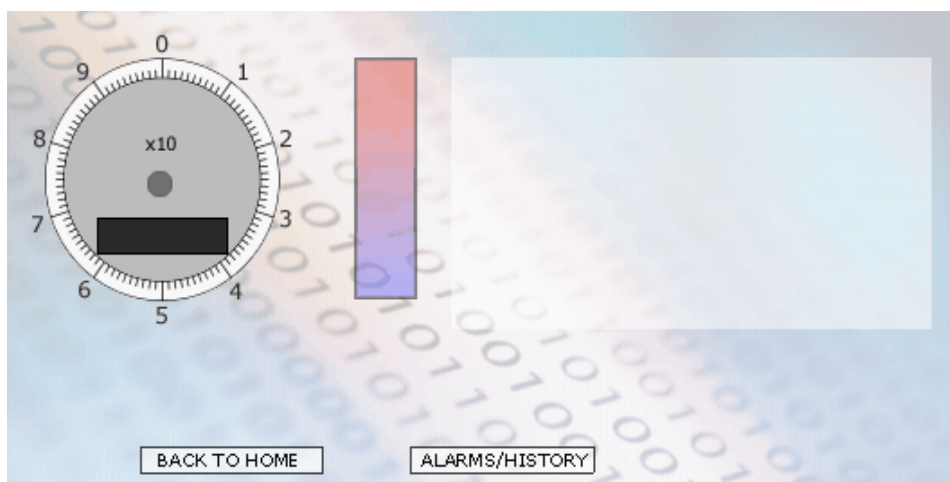
There is however a practical limit to the refresh rate of a web page; while the meta table is updated in real-time, a practical refresh rate for a web page is a few seconds. So while the meta table is always up-to-date, the data in the web page will typically lag behind. In addition, the web server and standard web pages are limited to a fairly static one-way display of data as there are no animated graphs or dials and they do not support any user input.

### 7.3.1 Real-time webbased presentation

In contrast, the web based graphical user interface, known as WebGUI, provides a true interactive interface for FieldCommander. It accepts user input to enable you to interact with your application. Dials, graphs and bar indicators can be defined and updated in real-time, as data is pushed directly from FieldCommander to the screen via the WebGUI. In effect, the values on the screen are updated in real-time. Data can be sent to the WebGUI directly from the stream by using a Viewport component or inserting meta tags in the WebGUI configuration file.

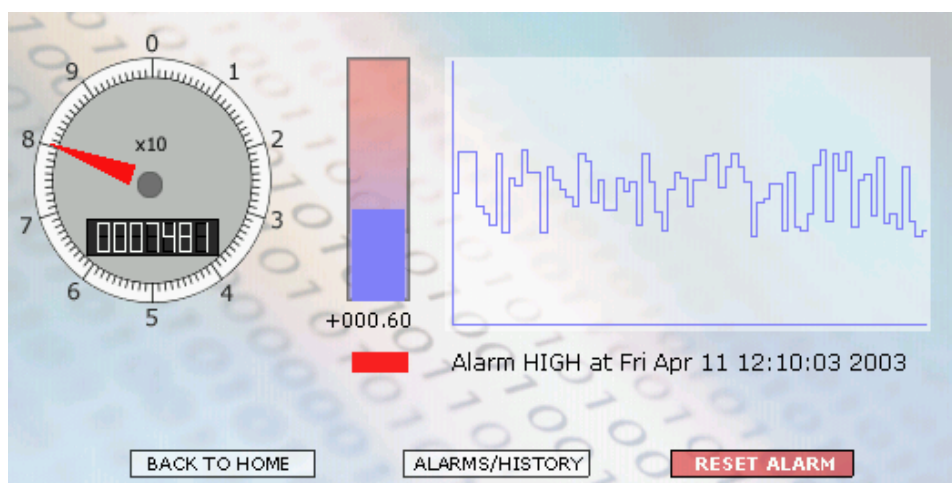
The WebGUI is a Java applet residing in FieldCommander and loaded into a browser window upon accessing the corresponding HTML pages that are served by FieldCommander's web server. Multiple instances of the WebGUI can exist in different browser windows on several network PC's simultaneously. The WebGUI itself consist of a background image with several dynamic components. The web page area covered by the applet, the background image that is used and the type, place and configuration of the dynamic components are determined by an xml-based configuration file. As the XML configuration file is merely a structured text-based document, you can create it using your favourite text editor and upload it to FieldCommander using FTP.

This picture shows the static background image:



### Dynamic components

The WebGUI components are updated in real-time and projected onto the background, as is shown here:



When a user opens an instance of the WebGUI, it contacts FieldCommander to pass the most recent data of meta tags, viewpoints and buffers that are used by the defined dynamic components. If the value of a meta tag or viewpoint is changed, it is passed to all instances of the WebGUI that are running and each WebGUI applet updates (redraws) all dynamic components that are registered to the specific meta tag or viewpoint.

Apart from the representation of real-time data, a user can also actively interact with FieldCommander and its running processes. Using a mouse to click on predefined 'hot' area's in the WebGUI applet, a user can generate an event which is handled in the `handleEvent()` function in the FieldCommander script.

It is also possible to directly change values of meta tags. This is also achieved by using 'hot' area's. Clicking with a mouse in a predefined 'hot' area causes the value of meta tag to be increased or decreased with a predefined value. The meta tag is immediately updated in FieldCommander's meta table.

### WebGUI requirements

The WebGUI applet requires the Java 2 Runtime Environment (JRE) to work. It is developed by Sun Microsystems and shipped with FieldCommander on the documentation CD, or freely available from [java.sun.com](http://java.sun.com). The applet connects to FieldCommander on TCP port 9734 and exchange data (see also paragraph 4.4).

### 7.3.2 WebGUI in a web page

The WebGUI applet is embedded into a web page and loaded from FieldCommander. The page file should the following code:

```
<HTML>
<HEAD>
</HEAD>
<BODY>

  <!-- put this in your HTML file to load WebGUI -->
  <APPLET CODE="WebGUI.class" JAVA_CODEBASE="." WIDTH="400" HEIGHT="300">
    <PARAM NAME=CODE VALUE="WebGUI.class">
    <PARAM NAME=CODEBASE VALUE=".">
    <PARAM NAME=ARCHIVE VALUE="/applet/WebGUI.jar">
    <PARAM NAME=TYPE VALUE="application/x-java-applet;version=1.4">
    <PARAM NAME=IPADDRESS VALUE="<!--#meta _IPADDRESS -->">
```

```

    <PARAM NAME=XMLCONFIG VALUE="/test.xml">
  </APPLET>
  <!-- end of WebGUI specific code -->

</BODY>
</HTML>

```

The <APPLET> block can be placed anywhere in a HTML page. Several values have to be changed according to your needs:

WIDTH	: width of the applet in pixels
HEIGHT	: height of the applet in pixels
XMLCONFIG	: name of the XML configuration file to use
IPADDRESS	: IP address of the FieldCommander the HTML page is installed; use the <code>_IPADDRESS</code> meta tag to make FieldCommander fills it out for you

When the page is loaded, the web browser recognizes the <APPLET> tags, spawns the Java Virtual Machine which will load and execute the given applet. The applet connects to FieldCommander on TCP port 9734 and exchange data.

### 7.3.3 WebGUI configuration

#### XML configuration file

The WebGUI is configured and created with a XML based configuration file. As the XML configuration file is merely a structured text-based document, you can create it using your favourite text editor and upload it to FieldCommander using FTP. The standard syntax for the XML configuration file looks like this:

```

<?xml version="1.0"?>
<fc_applet_definition>
  <background>bgpicture</background>
  <title>configtitle</title>
  <component type="comptype">
    <element type="eltype">nodetext</element>
  </component>
  <variable type="varsource">
    <element>nodetext</element>
  </variable>
</fc_applet_definition>

```

#### Background element

```
<background>bgpicture</background>
```

The `background` element assigns a picture to be used as background for the screen space allocated by the WebGUI applet. The picture is resized to fit the applet size. When no background picture is given, the background is gray.

`bgpicture` is the path which points to the file. The path is relative from the location of html file which spawns the WebGUI Applet. Supported file types are JPEG, GIF (with transparency) and PNG (with true alpha transparency).

#### Title element

```
<title>configtitle</title>
```

The `title` element assigns the name in `configtitle` to the WebGUI configuration for identification. The title is included for your convenience only. It is currently not used by the applet and it can be omitted.

### Component element

```
<component type="comptype">
  <element type="eltype">nodetext</element>
</component>
```

The `component` node holds a type attribute (see `comptype`) and a number of `element` nodes which are dependent of the type chosen.

comptype	description
"bar"	filled bar for level indication, thermometers, progress etc
"meter"	gauge needle for analog dials
"indicator"	round or rectangular lamp indicator
"leddisplay"	7 segment LED display
"text"	text label in various fonts, sizes and styles
"image"	picture image in JPEG, GIF or PNG format
"realtimetrend"	real-time scrolling trend graph with data from meta tag or viewpoint
"historictrend"	historic non scrolling trend graph with data from a buffer
"historictable"	historic table with data from a buffer
"hotrect"	rectangular area that reacts to mouse clicks

The element content can be either static ("constant") or dynamic ("variable"). This is defined in the `eltype` attributed of `element`.

eltype	description
"constant"	fixed value or text, defined in the element's <code>nodetext</code>
"variable"	dynamic content, source defined in the element's <code>nodetext</code>

Note that only elements printed in *italics* (see following paragraphs) support the "variable" type. When the `eltype` attribute is omitted, is it assumed to be constant type.

Refer to the Appendix E for the different graphical components that can be defined with the component element.

### Variable element

```
<variable type="vartype">
  <element>element name</element>
</variable>
```

The variable element is the mechanism to connect graphical components to sources in

FieldCommander. There are two types defined by `vartype` where the variable can be connected to; a meta tag and a viewpoint.

When connected to a meta tag, the meta value is sent to the WebGUI as soon as a meta tag is updated in FieldCommander. If the viewpoint component that is positioned in the stream receives a new data unit, it sends the value of the data element defined by the `element` subelement to the WebGUI.

Different graphical components can refer to one variable element. This means that one changed value in the FieldCommander can result in a number of graphical components being updated. When the value of a meta tag or viewpoint is changed, it is passed to all WebGUI's that are running. Each WebGUI applet updates (redraws) all dynamic components when new data arrives.

vartype	description
"meta tag"	fixed value or text, defined in the element's <code>nodetext</code>
"viewpoint"	dynamic content, source defined in the element's <code>nodetext</code>

The historic based components, which access the buffer, do not use the variable element.

## Appendix A: RS-232 communication

### The RS-232 standard

The RS-232 standard, or TIA/EIA-RS-232-C as is its official name, specifies the electrical characteristics of circuits between devices (modems and terminals) and gives names and numbers to the wires. The connections, topology and pin-out of the connectors are documented in the Installation Guide.

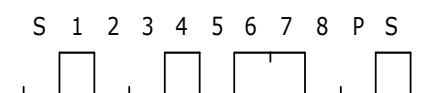
### The serial data

Each data frame in asynchronous communication such as RS-232, consists of a number of data bits (the actual data), start- and stop bits to control synchronization, a parity bit, together called *data frame*. Next to the data lines, RS-232 defines a set of lines to control the communication, the *control signals*.

### Data frame

The data frame consists of a number of data bits (the actual data to send or receive) plus start- and stop bits to control synchronization. Additionally a parity bit may be present which can determine incorrectly transferred bits.

Example: 1 start bit, 8 data bits, even parity and 1 stop bit



### Start bit

This first bit (always low) indicates the beginning of the data frame.

### Data bits

The data is the actual information that is transferred in either 5, 6, 7 or 8 bits. The order of transmission is from the least-significant to the most-significant bit. ASCII characters can be represented by either 7 or 8 bits, binary data bytes requires 8 bits.

### Parity

This optional bit makes it possible to do a check for errors during transmission. You can choose between None (no parity), Odd (all data bits and parity equal an odd number of '1' bits), Even (all data bits and parity equal an even number of '1' bits), Mark (parity is high) and Space (parity is low).

### Stop bits

The end of a frame is marked by a number of stop bits (always high). This can be either 1 or 2. If you use 5 data bits, setting 1 stop bit generates 1½ stop bit.

### Flow control

#### Hardware flow control

Hardware handshaking ("out-of-band" signaling) is the use of dedicated handshaking circuits to control the transmission of data. To summarize: DCE equipment normally uses DSR (Data Set Ready) as a main handshaking line to tell DTE that it is powered up and ready to control transmissions it is receiving.

The DTE uses DTR (Data Terminal Ready) as a main handshaking line. This is generally known as "DTR/DSR flow control". Another commonly used handshaking scheme uses CTS (Clear To Send) for DCE and also RTS (Request To Send) for DTE, called "RTS/CTS flow control".

By convention, the handshake wires carry a positive voltage when transmission is to be enabled, and a negative voltage when it is to be suspended.

#### *Software flow control*

When handshaking signals are sent as data along the data wires (TxD and RxD) instead of over dedicated signal lines, this is called software or "in-band" handshaking. The receiving device sends ASCII character DC3 (most commonly 19 decimal, 13 Hex) to the transmitting device when it wants to stop the transmitting device from sending characters. It sends ASCII character DC1 (17 decimal, 11H in most applications) when it wants the transmission to resume. This is often referred to as "XON/XOFF flow control."

### **Status signals**

FieldCommander is intended to acquire and send data over your serial lines. Besides the data bytes, it also determines the status of control and error signals.

#### *Control signals*

Next to the data lines, several other pins on the RS-232 connector are used to control the transmission which are stored along with the data:

#### **CTS - Clear to send**

This signal indicates that the serial device is ready to receive data. This signal is a response to the RTS (Request To Send) signal from the other device.

#### **RTS - Request to Send**

This signal indicates that the serial device is ready to send data.

#### **DTR - Data Terminal Ready**

This signal indicates that the serial device is ready to receive data. This signal is a response to the DSR (Data Set Ready) signal from the other device.

#### **DSR - Data set ready**

This signal indicates that the serial device is ready to send data.

#### **DCD - Data carrier detect**

Used by a modem to indicate the presence of a carrier signal.

#### **RI - Ring indicator**

Used by a modem to indicate that it is receiving a call.

### *Error signals*

The received serial data can suffer from a number of errors. FieldCommander catches various errors and stores them along with the data:

**OE - Overrun Error**

This signal is high when an overrun error occurs. An overrun exists when a new character has been received before the last character was read from the serial ports buffer, indicating data *might* be lost. It occurs when there is too much data for FieldCommander to process in the given period. You might improve performance by changing the FIFO trigger setting of the hardware port.

**PE - Parity Error**

This signal is high when a byte was received with an incorrect parity. This error occurs when there was an error on the line or the two serial devices under test have different parity settings.

**FE - Frame Error**

This signal is high when a framing error occurs. It occurs when the received data byte does not have valid stop bit(s). Most of the time this error is reported when the number of stop bits and/or data bits are incorrect or an error has occurred on the line.

**BRK - Break**

FieldCommander detected a break condition. A break is any period of a low signal on the receive line for longer than a normal character length time.

**Configuration**

All serial parameters, such as baud rate, data frame settings, flow control scheme and FIFO trigger level, are configurable per port through the script function *setLocalPort()*. The *FCscript Programmer's Guide* explains how to set the parameters.

## Appendix B: RS-485 communication

### The RS-485 standard

The RS-485 standard, or TIA/EIA-RS-485-A as is its official name, specifies the electrical characteristics of circuits between devices (modems and terminals) and gives names and numbers to the wires. The connections, topology and pin-out of the connectors are documented in Installation Guide.

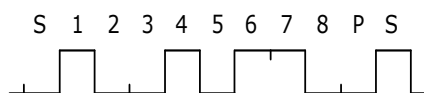
### The serial data

Each data frame in asynchronous communication such as RS-485, consists of a number of data bits (the actual data), start- and stop bits to control synchronization, a parity bit, together called *data frame*. Communication errors are detected and stored along with the data as *control signals*.

### Data frame

The data frame consists of a number of data bits (the actual data to send or receive) plus start- and stop bits to control synchronization. Additionally a parity bit may be present which can determine incorrectly transferred bits.

Example: 1 start bit, 8 data bits, even parity and 1 stop bit



#### Start bit

This first bit (always low) indicates the beginning of the data frame.

#### Data bits

The data is the actual information that is transferred in either 5, 6, 7 or 8 bits. The order of transmission is from the least-significant to the most-significant bit. ASCII characters can be represented by either 7 or 8 bits, binary data bytes requires 8 bits.

#### Parity

This optional bit makes it possible to do a check for errors during transmission. You can choose between None (no parity), Odd (all data bits and parity equal an odd number of '1' bits), Even (all data bits and parity equal an even number of '1' bits), Mark (parity is high) and Space (parity is low).

#### Stop bits

The end of a data frame is marked by a number of stop bits (always high). This can be either 1 or 2. If you use 5 data bits, setting 1 stop bit effectively generates 1½ stop bit.

### Flow control

FieldCommander is equipped with automatic flow control for RS-485. It automatically senses the direction of incoming data and switches its transmission direction accordingly. Therefore no handshaking signal (e.g. RTS signal) is necessary. This feature lets you simply and quickly build an RS-485 network with just two wires.

## Status signals

FieldCommander is intended to acquire and send data over your serial lines. Besides the data bytes, it also determines the status of various error signals. FieldCommander catches various error states and stores them along with the data.

### *Error signals*

#### **OE - Overrun Error**

This signal is high when an overrun error occurs. An overrun exists when a new character has been received before the last character was read from the serial port's buffer, indicating data *might* be lost. It occurs when there is too much data for FieldCommander to process in the given period. You might improve performance by changing the FIFO trigger setting of the hardware port.

#### **PE - Parity Error**

This signal is high when a byte was received with an incorrect parity. This error occurs when there was an error on the line or the two serial devices under test have different parity settings.

#### **FE - Frame Error**

This signal is high when a framing error occurs. It occurs when the received data byte does not have valid stop bit(s). Most of the time this error is reported when the number of stop bits and/or data bits are incorrect or an error has occurred on the line.

#### **BRK - Break**

FieldCommander detected a break condition. A break is any period of a low signal on the receive line for longer than a normal character length time.

## Configuration

All serial parameters, such as baud rate, data frame settings, flow control and FIFO trigger level, are configurable per port through the script function *setLocalPort()*. The *FCscript Programmer's Guide* explains how to set the parameters.

## Appendix C: Software editions

Features	Software edition		
	Standard	Advanced	Extended
<b>General</b>			
Scripting language for application programming (FCscript)			
Meta tags, max.	250	4000	4000
Relational SQL database			
Data processing components, max.	16	64	64
<b>User interface</b>			
Web based configuration and administration			
Custom dynamic HTML pages with SSI			
Custom dynamic PHP pages (FCphp)			
Realtime notifications to browser (WebNotify)			
Realtime graphic display in browser (WebGUI)			
Embedded browser with Java™ & virtual keyboard			
<b>Communication</b>			
Telephony functions			
Text messaging (SMS) in/out			
Multimedia messaging (MMS)		○	○
Audio playback and DTMF recognition		○	○
FTP client for file transfer			
E-mail client (SMTP) with HTML and attachments			
Export data to other FieldCommanders over ethernet and GPRS			
Import data from other FieldCommanders over ethernet			
XML export			
Camera support and imaging functions		○	○
<b>Network services</b>			
Web server (HTTP)			
Web server with SSL (HTTPS)			
FTP server for system administration			
Network time protocol (NTP) client/server			
DHCP/BOOTP			

: included in all models

○: depending on model, see product data sheet

## Appendix D: SQL syntax

### Types:

```
TEXT
NUMERIC
INTEGER
NONE
```

### Conditionals:

```
||
* / %
+ -
<< >> & |
< <= > >=
= == != <> IN
AND
OR
[ISNULL] [NOTNULL] [NOT] LIKE
[NOT] IN ( [,...] ) [NOT] BETWEEN x AND y
[conditional] ANY ( [,...] )
[conditional] ALL ( [,...] )
```

### Functions:

```
ABS | AVG | MAX | MIN | SUM | COUNT
LENGTH | ROUND | LOWER | UPPER | RANDOM
SUBSTR(string,start,length)
MAX(v1,v2,..) | MIN(v1,v2,..)
```

### Table Constraints:

```
[CONSTRAINT cname] {{UNIQUE|PRIMARY KEY}(col,...)|
REFERENCES table(col,...)}
```

### Column Constraints:

```
[CONSTRAINT cname] {[NOT] NULL|UNIQUE|PRIMARY KEY|
REFERENCES table(col,...) ON DELETE CASCADE}
```

### Command: ALTER TABLE

Description: Modifies table properties

```
ALTER TABLE table
  ADD COLUMN column type
ALTER TABLE table
  RENAME TO newtable
```

### Command: CREATE TABLE

Description: Creates a new table

```
CREATE <[TEMPORARY|TEMP]> TABLE table (
  column type
  [ NOT NULL ] [ UNIQUE ] [ DEFAULT value ] [ CHECK (expr) ]
  [ PRIMARY KEY ( column [, ...] ) ] [AUTOINCREMENT]
  [, table constraint ]
)
```

**Command: CREATE TRIGGER (not SQL92)**

Description: Creates a new trigger

```
CREATE <[TEMPORARY|TEMP]> TRIGGER name { BEFORE | AFTER }
  {DELETE| INSERT | UPDATE [OF (col,...)] [OR ...] }
  ON {table|view} FOR EACH ROW
  [ WHEN (condition) ]
  [<!EXECUTE PROCEDURE func ( arguments )>| pl/sql block]
```

**Command: CREATE INDEX (not SQL92)**

Description: Constructs a secondary index

```
CREATE [ UNIQUE ] INDEX index_name ON table
  ( column [ASC|DESC][,...] )
```

**Command: CREATE VIEW**

Description: Constructs a virtual table

```
CREATE <[TEMPORARY|TEMP]> VIEW view AS select query
```

**Command: DROP**

Description: Removes existing objects from database

```
DROP TABLE name
DROP VIEW name
DROP INDEX name
DROP TRIGGER name
```

**Command: INSERT**

Description: Inserts new rows into a table

```
INSERT [OR REPLACE] INTO table [ ( column [, ...] ) ]
  { VALUES ( expression [, ...] ) | SELECT query }
```

**Command: REPLACE**

Description: Inserts rows into a table

```
REPLACE INTO table [ ( column [, ...] ) ]
  { VALUES ( expression [, ...] ) | SELECT query }
```

**Command: UPDATE**

Description: Replaces values of columns in a table

```
UPDATE table SET col = expression [,...]
  [ WHERE condition ]
```

**Command: DELETE**

Description: Removes rows from a table

```
DELETE FROM table [ WHERE condition ]
```

**Command: SELECT query**

Description: Retrieve rows from a table or view

```
SELECT [ ALL | DISTINCT [ ON ( expression [, ...] ) ] ]
  expression [ <[AS]> name ] [,...]
  [ INTO [ TEMPORARY | TEMP ] [ TABLE ] new_table ]
  [ FROM {table | (select query)} [ alias ] [,...] ]
  [NATURAL] [LEFT | RIGHT | FULL] [OUTER | INNER | CROSS] JOIN table alias
  {ON condition | USING(col1,col2,...)} ]
  [ WHERE {condition | EXISTS (correlated subquery)} ]
  [ GROUP BY column [,...] ]
```

```
[ HAVING condition [,...] ]  
[ { UNION [ ALL ] | INTERSECT | EXCEPT } select ]  
[ ORDER BY {column | int} [ ASC | DESC | USING operator ] [,...] ]  
[ FOR UPDATE [ OF class_name [,...] ] ]  
LIMIT { count | ALL } [ { OFFSET | , } start ]
```

**Command: VACUUM**

Description: Frees unused space in database

```
VACUUM
```

**Command: BEGIN TRANSACTION (not SQL92)**

Description:

Any command that changes the database (basically, any SQL command other than SELECT) will automatically start a transaction if one is not already in effect. Automatically started transactions are committed at the conclusion of the command.

Transactions can be started manually using the BEGIN command. Such transactions usually persist until the next COMMIT or ROLLBACK command. But a transaction will also ROLLBACK if the database is closed or if an error occurs and the ROLLBACK conflict resolution algorithm is specified. See the documentation on the ON CONFLICT clause for additional information about the ROLLBACK conflict resolution algorithm.

Deferred means that no locks are acquired on the database until the database is first accessed. Thus with a deferred transaction, the BEGIN statement itself does nothing. Locks are not acquired until the first read or write operation. The first read operation against a database creates a SHARED lock and the first write operation creates a RESERVED lock. Because the acquisition of locks is deferred until they are needed, it is possible that another thread or process could create a separate transaction and write to the database after the BEGIN on the current thread has executed. If the transaction is immediate, then RESERVED locks are acquired on all databases as soon as the BEGIN command is executed, without waiting for the database to be used. After a BEGIN IMMEDIATE, you are guaranteed that no other thread or process will be able to write to the database or do a BEGIN IMMEDIATE or BEGIN EXCLUSIVE. Other processes can continue to read from the database, however. An exclusive transaction causes EXCLUSIVE locks to be acquired on all databases. After a BEGIN EXCLUSIVE, you are guaranteed that no other thread or process will be able to read or write the database until the transaction is complete. The default behavior is a deferred transaction.

```
BEGIN [ DEFERRED | IMMEDIATE | EXCLUSIVE ] [TRANSACTION [name]]  
END [TRANSACTION [name]]  
COMMIT [TRANSACTION [name]]  
ROLLBACK [TRANSACTION [name]]
```

## Appendix E: WebGUI components

Type definitions for the graphical components:

STRING	= string of characters, max length 255
INTEGER	= integer value
FLOAT	= floating point value
HEXRGB	= color in hexadecimal RGB notation as in HTML (#RRGGBB)
PERC	= percentage, integer value ranging from 0 to 100
FILEREF	= file name with path in FieldCommander from web root
URL	= resource reference, absolute (with <a href="http://">http://</a> ) or relative from web root
BOOLEAN	= 0 (off/no/disabled) or 1 (on/yes/enabled)

If a supplied value in a the configuration file does not conform to the type definition of the element then the defined element is ignored.

Refer to the `component` element for more information.

### Bar

The bar is a flexible rectangle on a defined location on the screen.

The bar has several properties, each defined by a separate xml element:

name	= STRING used as identification for the component, for example by "refresh"	
x-pos	= INTEGER defines the x-position of the left upper corner of the component in regard to the left upper corner of the applet	
y-pos	= INTEGER defines the y-position of the left upper corner of the component in regard to the left upper corner of the applet	
width	= INTEGER width of the bar in pixels	
height	= INTEGER height of the bar in pixels	
direction	= "up" / "down" / "left" / "right" direction in which the bar expands; see the image on the right	
color	= HEXRGB color of the bar	
visible	= BOOLEAN defines if the bar is visible, 0 is invisible, 1 is visible	
scale_start	= INTEGER defines how the value element is interpreted. A scale_start of 0 means that all values below 0 do not show anything and that values larger than 0 make the bar expand.	
scale_end	= INTEGER defines how the value element is interpreted. A scale_end of 100 means that all values larger then 100 show a fully expanded bar and all values below 100 show a bar between the scale_start and scale_end.	
value	= INTEGER / FLOAT constant value or name of a variable component. Refer to the variable component for	

more information

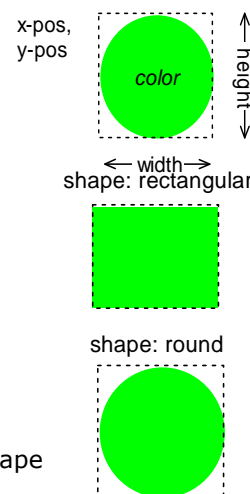
Example:

```
<component type="bar">
  <name>tank 1</name>
  <x-pos>60</x-pos>
  <y-pos>60</y-pos>
  <width>20</width>
  <height>100</height>
  <direction>up</direction>
  <color>#FF0000</color>
  <visible>1</visible>
  <scale_start>0</scale_start>
  <scale_end>100</scale_end>
  <value type="variable">tank1_variable</value>
</component>
```

### Indicator

The indicator is an oval or rectangle that is shown when the state is not zero. If the state is zero nothing is shown.

name	= STRING	used as identification for the component, for example by "refresh"
x-pos	= INTEGER	defines the x-position of the left upper corner of the component in regard to the left upper corner of the applet
y-pos	= INTEGER	defines the y-position of the left upper corner of the component in regard to the left upper corner of the applet
width	= INTEGER	width of the bar in pixels
height	= INTEGER	height of the bar in pixels
shape	= "round" / "rectangular"	a round shape results in a circle or oval indicator, a rectangular shape results in a square or rectangle
color	= HEXRGB	color of the indicator
blink	= BOOLEAN	0 stands for no blinking, 1 stands for blinking. When "visible" is true, the indicator blinks with an interval defined by blink_time
blink_time	= INTEGER	interval time in milliseconds for the blinking mode of the indicator
visible	= BOOLEAN	constant boolean value or the name of a variable component. Refer to the variable component for more information. When the variable is zero, the indicator is not visible, if the variable is larger than zero the indicator is shown.



Example:

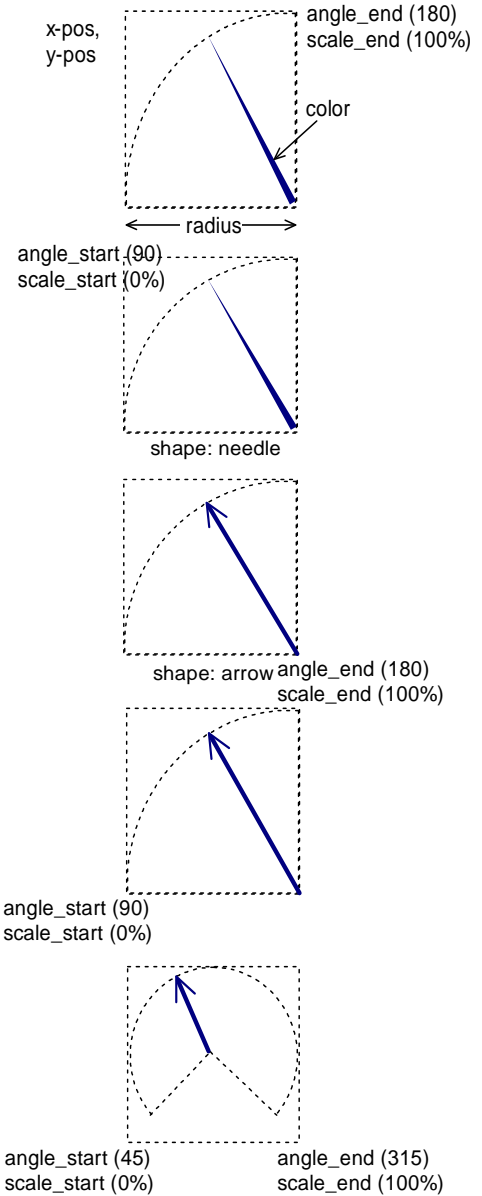
```
<component type="indicator">
  <name>alert 1</name>
  <x-pos>60</x-pos>
  <y-pos>80</y-pos>
  <width>20</width>
  <height>20</height>
  <shape>round</shape>
  <color>#FF0000</color>
  <blink>1</blink>
```

```
<blink_time>200</blink_time>
<visible type="variable">alert1_variable</visible>
</component>
```

**Meter**

A meter is a pointer that uses a angle to represent a value. The pointer will always go clockwise from the angle start to the angle end position.

- name** = STRING  
used as identification for the component, for example by "refresh"
- x-pos** = INTEGER  
defines the x-position of the left upper corner of the component in regard to the left upper corner of the applet
- y-pos** = INTEGER  
defines the y-position of the left upper corner of the component in regard to the left upper corner of the applet
- radius** = INTEGER  
length of the needle or arrow
- shape** = "needle" / "arrow"  
shape of the pointer
- color** = HEXRGB  
color of the pointer
- visible** = BOOLEAN  
defines if the meter is visible. 0 is invisible, 1 is visible
- scale\_start** = PERC  
defines how the value element is interpreted. A scale\_start of 0 means that all values below 0 will show the pointer at the angle\_start position, values larger than 0 make the pointer move clockwise towards the angle\_end position.
- scale\_end** = PERC  
defines how the value element is interpreted. A scale\_end of 100 means that all values larger then 100 show the pointer at the angle\_end position.
- angle\_start** = INTEGER  
position in degrees were the pointing device should be if the value is smaller or equal to the scale\_start value.
- angle\_end** = INTEGER  
position in degrees were the pointing device should be if the value is larger or equal to the scale\_end value.
- value** = INTEGER / FLOAT  
constant value or name of a variable component. Refer to the variable component for more information



Example:

```
<component type="meter">
  <name>meter 1</name>
  <x-pos>60</x-pos>
  <y-pos>100</y-pos>
```

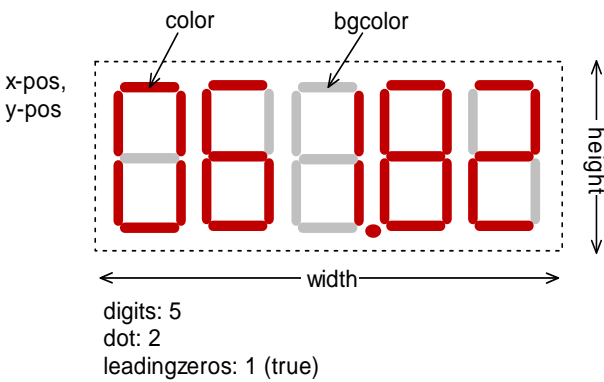
```

    <radius>20</radius>
    <shape>needle</shape>
    <color>#FF0000</color>
    <visible type="variable">meter_visible_variable</visible>
    <scale_start>0</scale_start>
    <scale_end>100</scale_end>
    <angle_start>45</angle_start>
    <angle_end>315</angle_end>
    <value type="variable">meter1_variable</value>
  </component>

```

### LED display

A led display is a representation of a integer or float with 7 segment led displays.

<p><b>name</b> = STRING used as identification for the component, for example by "refresh"</p> <p><b>x-pos</b> = INTEGER defines the x-position of the left upper corner of the component in regard to the left upper corner of the applet</p> <p><b>y-pos</b> = INTEGER defines the y-position of the left upper corner of the component in regard to the left upper corner of the applet</p> <p><b>width</b> = INTEGER width of the led display in pixels</p> <p><b>height</b> = INTEGER height of the led display in pixels</p> <p><b>color</b> = HEXRGB color of the active led segments</p> <p><b>bgcolor</b> = HEXRGB color of the inactive led segments</p> <p><b>digits</b> = INTEGER number of digits shown</p> <p><b>dot</b> = INTEGER location of the dot from the right, effectively the number of decimals shown</p> <p><b>leadingzeros</b> = BOOLEAN defines if the left side of the led display is filled with zeros; 0 is false, 1 is true</p> <p><b>visible</b> = BOOLEAN defines if the led display is visible; 0 is invisible, 1 is visible</p> <p><b>value</b> = INTEGER / FLOAT constant value or name of a variable component. Refer to the variable component for more information</p>	 <p>digits: 5 dot: 2 leadingzeros: 1 (true)</p>
---	---

Example:

```

<component type="leddisplay">
  <name>display 1</name>
  <x-pos>60</x-pos>
  <y-pos>140</y-pos>
  <width>100</width>
  <height>20</height>
  <color>#FF0000</color>
  <bgcolor>#333333</bgcolor>
  <digits>5</digits>
  <dot>2</dot>

```

```

    <leadingzeros>1</leadingzeros>
    <visible>1</visible>
    <value type="variable">display1_variable</value>
  </component>

```

## Text

With a text custom text can be displayed anywhere in the applet window.

font: "sansserif" style: "italic" size: 10

x-pos,  
y-pos

Your text string

**name** = STRING  
used as identification for the component, for example by "refresh"

**x-pos** = INTEGER  
defines the x-position of the left upper corner of the component in regard to the left upper corner of the applet

**y-pos** = INTEGER  
defines the y-position of the left upper corner of the component in regard to the left upper corner of the applet

**font** = "monospaced" / "serif" / "sansserif" / <font name>\*

**style** = "plain" / "bold" / "italic" / "bold\_italic"  
font style

**size** = INTEGER  
font size in points

**color** = HEXRGB  
font color

**visible** = BOOLEAN  
defines if the led display is visible; 0 is invisible, 1 is visible

**string** = STRING  
constant string or name of a variable component. Refer to the variable component for more information

Monospaced	plain	10 points
SansSerif	bold	16 points
Serif	italic	
comic sans ms	bold_italic	30 points

*note that this example has been scaled*

\* monospaced, serif and sansserif are mapped to a physical fonts available on the client system. When supplying a different font name, it must be installed on the client system. True-type is available on all platforms, other font technologies are platform dependent.

Example:

```

<component type="text">
  <name>tank 1</name>
  <x-pos>120</x-pos>
  <y-pos>60</y-pos>
  <font>monospaced</font>
  <style>plain</style>
  <size>14</size>
  <color>#FF0000</color>
  <visible>1</visible>
  <string type="variable">text1_variable</string>
</component>

```

## Image

An image component defines one or more images that can be shown depending on the value of a variable.

**name** = STRING

x-pos,  
y-pos



<i>x-pos</i>	used as identification for the component, for example by "refresh" = INTEGER defines the x-position of the left upper corner of the component in regard to the left upper corner of the applet
<i>y-pos</i>	= INTEGER defines the y-position of the left upper corner of the component in regard to the left upper corner of the applet
<i>width</i>	= INTEGER width of the image in pixels
<i>height</i>	= INTEGER height of the image in pixels
<i>image</i>	= FILEREF file reference to an image that has to be displayed. Multiple images can be specified, each with another value assigned (see example)
<i>stretch</i>	= BOOLEAN defines whether the image should be scaled to fit the given width and height; 0 means that original image size is preserved, 1 means scale to fit.
<i>visible</i>	= BOOLEAN defines if the image is visible; 0 is invisible, 1 is visible
<i>value</i>	= INTEGER constant value or name of a variable component. Refer to the variable component for more information. The defined value or the value of the variable component defines which image is shown.

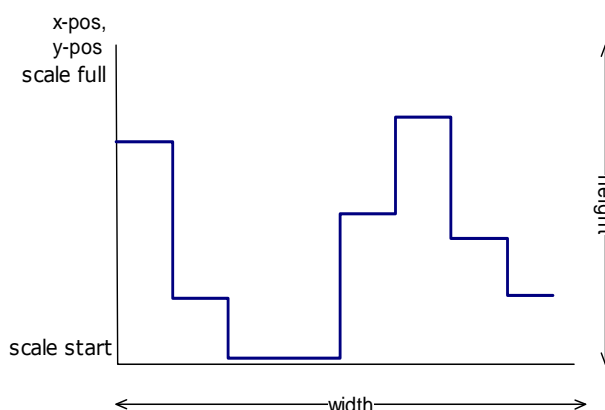
Example:

```
<component type="image">
  <name>image 1</name>
  <x-pos>120</x-pos>
  <y-pos>100</y-pos>
  <width>100</width>
  <height>100</height>
  <image value="0">image1.gif</image>
  <image value="1">image2.gif</image>
  <image value="2">image3.gif</image>
  <visible>1</visible>
  <value type="variable">image1_variable</value>
</component>
```

### Real-time trend

The Real-time trend shows a graph which is initially empty and as time goes by shows the trend of the defined variable. As the graph has a transparent background, you can overlap multiple trends.

<i>name</i>	= STRING used as identification for the component, for example by "refresh"
<i>x-pos</i>	= INTEGER defines the x-position of the left upper corner of the component in regard to the left upper corner of the applet
<i>y-pos</i>	= INTEGER defines the y-position of the left upper corner of the component in regard to the left upper corner of the applet
<i>width</i>	= INTEGER



height	width of the trend in pixels = INTEGER
color	height of the trend in pixels = HEXRGB
scale_start	color of the trend and the x- and y-axis = INTEGER
scale_end	defines the lower limit of the trend = INTEGER
valuecount	defines the upper limit of the trend = INTEGER
interval	defines the number of values that is drawn on the x-axis = INTEGER
value	defines how often the trend is updated, time in milliseconds = INTEGER / FLOAT
	name of a variable component. Refer to the variable component for more information.

Example:

```
<component type="realtimetrend">
  <name>trend 1</name>
  <x-pos>200</x-pos>
  <y-pos>20</y-pos>
  <width>100</width>
  <height>100</height>
  <color>#FF0000</color>
  <scale_start>10</scale_start>
  <scale_end>120</scale_end>
  <valuecount>50</valuecount>
  <interval>500</interval>
  <visible>1</visible>
  <value type="variable">trend1_variable</value>
</component>
```

### Historic table

A historic table will show the last N number of data units from a specified data buffer.

name	= STRING used as identification for the component, for example by "refresh"
x-pos	= INTEGER defines the x-position of the left upper corner of the component in regard to the left upper corner of the applet
y-pos	= INTEGER defines the y-position of the left upper corner of the component in regard to the left upper corner of the applet
width	= INTEGER width of the table in pixels
height	= INTEGER height of the table in pixels
column element	= STRING element of data unit in the data buffer that should be displayed by the column
	header = STRING

time	data
2003-04-09 11:10:25.708	+007.40
2003-04-09 11:10:35.708	+007.60
2003-04-09 11:10:55.708	+007.10
2003-04-09 11:11:05.708	+000.70
2003-04-09 11:11:35.708	+003.30
2003-04-09 11:11:45.708	+009.70
2003-04-09 11:11:55.708	+000.50
2003-04-09 11:12:15.708	+007.70
2003-04-09 11:12:25.708	+000.00
2003-04-09 11:12:35.707	+003.80
2003-04-09 11:12:45.708	+007.40

x-pos, y-pos

width

height

text that is displayed in the header above the column

**unitcount** = INTEGER  
the number of units that is shown in the table. The oldest units are on top.

**visible** = INTEGER  
defines if the table is visible, 0 is invisible, 1 is visible

**source** = STRING  
name of the databuffer were the data should be fetched from

Example:

```
<component type="historictable">
  <name>table1</name>
  <x-pos>300</x-pos>
  <y-pos>120</y-pos>
  <width>200</width>
  <height>200</height>
  <column>
    <element>timestamp</element>
    <header>time</header>
  </column>
  <column>
    <element>data</element>
    <header>data</header>
  </column>
  <unitcount>20</unitcount>
  <visible>1</visible>
  <source>mybuffer</source>
</component>
```

### Historic trend

The historic trend is a graphical presentation of the last N data units from a data buffer.

**name** = STRING  
used as identification for the component, for example by "refresh"

**x-pos** = INTEGER  
defines the x-position of the left upper corner of the component in regard to the left upper corner of the applet

**y-pos** = INTEGER  
defines the y-position of the left upper corner of the component in regard to the left upper corner of the applet

**width** = INTEGER  
width of the trend in pixels

**height** = INTEGER  
height of the trend in pixels

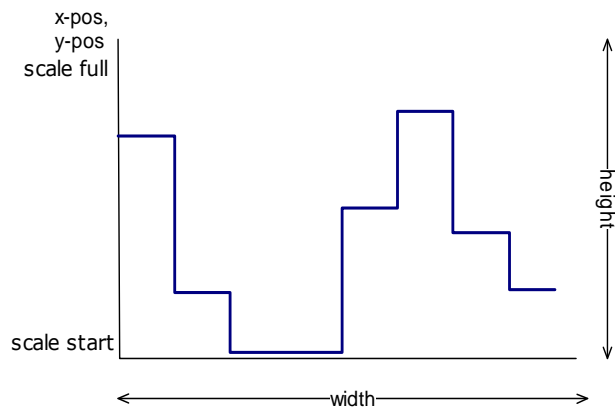
**color** = HEXRGB  
color of the trend and the x- and y-axis

**visible** = INTEGER  
defines if the table is visible, 0 is invisible, 1 is visible

**scale\_start** = INTEGER  
defines the lower limit of the trend

**scale\_end** = INTEGER  
defines the upper limit of the trend

**unitcount** = INTEGER



source	defines the number of data units that are read by the specified buffer = STRING
element	the name of a databuffer = STRING the name of a data element of the data type from the specified buffer, this data element is the data that is used for the y-axis of the trend.

**Example:**

```
<component type="historictrend">
  <name>trend2</name>
  <x-pos>300</x-pos>
  <y-pos>320</y-pos>
  <width>200</width>
  <height>200</height>
  <color>#FF0000</color>
  <scale_start>10</scale_start>
  <scale_end>20</scale_end>
  <unitcount>20</unitcount>
  <visible>1</visible>
  <source>mybuffer</source>
  <element>data</element>
</component>
```

**HotRect**

A HotRect defines "hot" area's in the WebGUI where mouse clicks are caught and actions can be initiated. There are four different types of action: refresh, hyperlink, increase/decrease a variable or trigger an event in FieldCommander.

name	= STRING used as identification for the component, for example by "refresh"
x-pos	= INTEGER defines the x-position of the left upper corner of the component in regard to the left upper corner of the applet
y-pos	= INTEGER defines the y-position of the left upper corner of the component in regard to the left upper corner of the applet
width	= INTEGER width of the hotrect area in pixels
height	= INTEGER height of the hotrect area in pixels
active	= BOOLEAN defines if the hotrect is active or not, 0 is not active, 1 is active
action	= "refresh" / "variable" / "hyperlink" / "trigger" defines the action for the hotrect: <ul style="list-style-type: none"> <li>- refresh: refresh the defined components, this is mostly used in combination with the historic components</li> <li>- variable: increase/decrease change the value of a variable (meta tag)</li> <li>- hyperlink: opens the defined URL in the browser (see also "target")</li> <li>- trigger: generate an event in the FieldCommander that can be handled in the eventHandle() function of the custom script.</li> </ul>
target	= "_blank" / "_self" / "_parent" / "_top" target to open the hyperlink for the browser window <ul style="list-style-type: none"> <li>- _blank: open link in new browser window</li> <li>- _self: open link in the current window and frame, this is default</li> <li>- _parent: open link to replace the parent frame set</li> <li>- _top: open link in the topmost frame set, so it fills the entire window</li> </ul>
value	= INTEGER

	only defined for variable mode: defines the value or variable that has to be increased or decreased. If the variable is of the type meta tag the new value is send through to the FieldCommander
left	<p>= INTEGER / FLOAT / STRING / URL</p> <p>defines the action that should be initiated when a left mouse click:</p> <ul style="list-style-type: none"> <li>- refresh: component name of component to be updated / refreshed. The left element can be specified multiple times to update multiple components at once</li> <li>- variable: negative or positive integer of float value. If the value is positive the assigned variable is increased, if the value is negative the assigned variable is decreased. Multiple clicks in the defined trigger_time results in multiple increases or decreases at once</li> <li>- hyperlink: a valid absolute (starting with <a href="http://">http://</a>) or relative URL</li> <li>- trigger: integer value between 0 and 1023. These values can be used in the handleEvent function as event source identifiers to catch witch event is generated</li> </ul>
right	<p>= INTEGER / FLOAT / STRING /URL</p> <p>defines the action that should be initiated when a right mouse click. Note that you cannot right-click on a touchscreen!:</p> <ul style="list-style-type: none"> <li>- refresh: component name of component to be updated / refreshed. The left element can be specified multiple times to update multiple components at once</li> <li>- variable: negative or positive integer of float value. If the value is positive the assigned variable is increased, if the value is negative the assigned variable is decreased. Multiple clicks in the defined trigger_time results in multiple increases or decreases at once</li> <li>- hyperlink: a valid absolute (starting with <a href="http://">http://</a>) or relative URL</li> <li>- trigger: integer value between 0 and 1023. These values can be used in the handleEvent function as event source identifiers to trap the event that is generated</li> </ul>
trigger_time	<p>= INTEGER</p> <p>the trigger time is only used by the variable option. It is a time defined in milliseconds, if the user clicks twice or more in the specified time the assigned variable is increased or decreased a multiple of the defined value instead of increasing or decreasing it a multiple times with the defined value.</p>

## Examples:

```

<!-- increase alert_level on a left click, decrease on a right click -->
<component type="hotrect">
  <name>increase_decrease</name>
  <x-pos>100</x-pos>
  <y-pos>200</y-pos>
  <width>20</width>
  <height>20</height>
  <active>1</active>
  <action>variable</action>
  <value type="variable">alert_level</value>
  <left>1</left>
  <right>-1</right>
  <trigger_time>350</trigger_time>
</component>

<!-- link to absolute url on left click, relative url on right -->

<component type="hotrect">
  <name>link1</name>
  <x-pos>200</x-pos>
  <y-pos>100</y-pos>
  <width>20</width>
  <height>20</height>

```

```
<active>1</active>
<action>hyperlink</action>
<left>http://www.mydomain.com</left>
<right>/index.html</right>
</component>

<!-- refresh component1 on a left click, component2 and component3 on a right
click -->
<component type="hotrect">
  <name>refresh1</name>
  <x-pos>200</x-pos>
  <y-pos>100</y-pos>
  <width>20</width>
  <height>20</height>
  <active>1</active>
  <action>refresh</action>
  <left>component1</left>
  <right>component2</right>
  <right>component3</right>
</component>

<!-- trigger event 1 on a left click and event 2 on a right click -->
<component type="hotrect">
  <name>trigger</name>
  <x-pos>200</x-pos>
  <y-pos>100</y-pos>
  <width>20</width>
  <height>20</height>
  <active>1</active>
  <action>trigger</action>
  <left>1</left>
  <right>2</right>
</component>
```

## Glossary

applet	Java programs that are run from the browser, such as WebNotify and WebGUI.
application	The combination of a your custom script, web pages and possibly a database that enables FieldCommander to process data, manage events and visualize information.
ASCII	(American Standard Code for Information Interchange) A code for representing characters as numbers, with each letter assigned a number from 0 to 255.
component	Functional block in FieldCommander's data processing system. Several components together form a logical network through which a stream of data units flows.
condition	Specifies a data state or occurrence which results in a logical "true" or "false". Conditions can be combined into an expression, enabling you to filter data and trigger events to meet your application's needs.
config tables	A system of databases that contain tables. These tables contain data that is used in a script. With the Configuration tables, you can edit the data used in a script, without altering the script itself.
data type	Specifies the type of data in a data unit, e.g. DT_RS, DT_AF or DT_MB or DT_SMS.
data unit	Data structures used by FieldCommander. The carrier of information.
database	A structured data file. FieldCommander uses databases to store tables with data.
debug	Troubleshooting an application or script by reporting the status of line-per-line processing of (programming) code.
DNS	(Domain Name System) The name resolution system that lets users locate computers on a Unix network or the Internet (TCP/IP network) by domain name.
DTMF	(Dual-Tone MultiFrequency) The type of audio signals that are generated when you press the buttons on a touch-tone telephone. Used in IVR applications.
element	A "field" in a data unit. FieldCommander enables you to set condition rules to elements containing either a numeric value, a date/time indication, a string or a flag state.
event	An action or occurrence detected by a program. FieldCommander manages events in a central handleEvent() routine in the script. It allows you to respond to specified situations that occur during data processing.
event driven	A program designed and optimized to respond to events. FieldCommander can be set to work in an event-driven way by using the sleep() command. In this mode, it continuously monitors for events to occur and to respond to them immediately.
event manager	Program routine to handle and process events. FieldCommander's central handleEvent() function routine, enables you to process events in real-time the way you want.
expression	A logical combination of one or more conditions using logical (Boolean) syntax.
FCphp	The extensions of FieldCommander to the well-known PHP language.
FCscript	The scripting language used by FieldCommander, based on Javascript.
firmware	Operating System software of FieldCommander.
flag	Bit value (1 or 0) representing a logical "high" or "low". In FieldCommander flag are used to indicate the (error) state of a serial line. Flags are set in the Control element of the DT_RS data unit.
frame	A partial stream of data units that together form a logical "block" of coherent information, containing one or more fields.

---

FTP	(File Transfer Protocol) The protocol used on the Internet for sending and receiving files.
HTML	(HyperText Markup Language) The common language used to create documents on the World Wide Web.
IVR	(Interactive Voice Response) An automated telephone information system that speaks to the caller with a combination of fixed voice menus and realtime data from databases. The caller responds by pressing digits on the telephone (see DTMF).
Java	A programming language designed to develop applications, especially ones for the Internet, that can operate on different platforms.
Javascript	A popular scripting language which forms the basis of FCscript.
meta table	Internal database of FieldCommander, linking meta tags to meta values.
meta tag	A meta tag is linked to a meta value in FieldCommander's meta table. Meta tags read and written from the script and inserted into HTML pages to display dynamic content.
meta value	String value linked to a meta tag in FieldCommander's meta table.
MMS	(Multimedia Message Service) A service for sending rich messages with pictures and text to mobile G.M./GPRS phones.
network	Several components connected together to form a logical diagram. A picture of the network of components visualizes the concept of data processing by FieldCommander.
NTP	(Network Time Protocol) An Internet standard protocol that assures accurate synchronization to the millisecond of computer clock times in a network of computers.
PHP	(PHP Hypertext Preprocessor) A scripting language used to create dynamic web pages.
poll	Actively request the status of a device on predetermined intervals, by using a "question and answer" style of communication.
profile	A user account in FieldCommander with access permissions.
script	A text file, written in the FCscript language, with your application which is interpreted and executed by FieldCommander.
SMS	(Short Message Service) A service for sending text messages of up to 160 characters to mobile G.M. phones.
SSL	(Secure Sockets Layer) The leading security protocol on the Internet. The web server sends a certificate to the browser, which is used to authenticate the web site.
SQL	(Structured Query Language) A language used to interrogate and process data in a relational database.
stream	A stream represents the flow of data units in FieldCommander through a network of components. The network offers a structured setup to process acquired data and convert it into usable information to present on a web page.
string	A pattern or series of grouped characters.
table	A data matrix consisting of a range of rows and columns
TCP/IP	(Transmission Control Protocol/Internet Protocol) The suite of communications protocols used to connect hosts on the Internet or a network.
time stamp	An element value representing time and date of processing. FieldCommander adds times tamps to data units with microsecond resolution.
trigger condition	Conditional value on an element of a data unit to make a component react to a specified situation. Trigger conditions can be set to multiple elements in a data unit by the use of expressions.

---

user profile	Account which defined user-bound rights on FieldCommander, such as HTTP and FTP access and permissions to view/change the System configuration.
WebGUI	Web based graphic user interface, a feature in FieldCommander to present information in the form of real-time graphics instead of plain text.
WebNotify	Web based notification system, a feature in FieldCommander to send (push) information to a web browser in real-time.
web interface	The web based System configuration panel of FieldCommander.
web server	Service to make web pages available on a computer network or the Internet. You can use FieldCommander's web server to host your own custom build web pages to serve your application's needs.
XML	(Extensible Markup Language) XML is the universal format for structured documents and data on the Web.

## Index

ASCII Frame Converter	11, 48, 50
audio	7, 56, 57, 71, 86
authentication	7, 24, 25, 29, 30, 35, 36
bootp	20, 71
Buffer component	40, 49, 54
camera	7, 57, 71
certificate	16, 27, 38, 87
clock timer	53
conditions	6, 49, 50, 52, 53, 86, 87, 91
control signals	10, 47, 66, 67, 69
data bits	10, 47, 66, 68-70
data processing	8, 12-14, 40, 47, 48, 53, 71, 86, 87
data string	48
Data trigger	50, 53
data types	48
data units	11, 40-42, 47-56, 81-83, 86, 87
debug	19, 26, 37, 86
DHCP	20, 71
DNS lookup	21
DT_AF	11-13, 48, 50, 51, 86
DT_MB	86
DT_RS	10, 12, 41, 47-51, 86
DT_SMS	48, 49, 86
DTMF	56, 57, 71, 86, 87
Ethernet	6, 9, 20, 54, 71
event callback	53
event driven	12, 13, 53, 86
event handling	7, 8, 14, 47, 50, 53, 54
event sources	53
Export	9, 51, 55, 56, 71
FCphp	5, 6, 8, 15, 29, 40, 42-44, 59, 71, 86
FCscript	5, 6, 8, 9, 18, 35-37, 40, 42-45, 48, 53, 54, 58, 59, 68, 70, 71, 86, 87
file management	24, 34
Filter	11-13, 47, 49, 52, 86
firewall	38
firmware updates	35
floating point	75
FTP client	9, 15, 34, 35, 38, 55, 71
FTP server	6, 9, 24, 29, 34, 38, 39, 43, 55, 71
Gate	49, 50, 52
gateway	20, 54
GPRS	7, 71, 87
GSM	7, 9, 56
Host name	18, 21
HTML pages	8, 58, 61, 71, 87
imaging	7, 57, 71
Import	51, 55, 71
interval timer	12, 13, 53
IP addressing	15, 20
Java applet	61
LocalPort	48, 49
log file	19, 26, 37, 58
login	16, 17
mail server	23
meta tags	11-14, 40-42, 46, 58, 61, 62, 71, 87
MMS	7, 9, 56, 71, 87

---

password	15, 17, 18, 25, 30, 34, 36-38
permissions	15, 17, 28-30, 35, 37, 87, 88
PHP engine	37, 58
PHP pages	8, 15, 37, 71
ping	20
poll	10, 53, 54, 87
Reboot	18, 29, 38, 39
relational database	40, 87
Remote Port	51, 56
RS-485	69
RTS/CTS	67
scripting	5-7, 71, 86, 87
security	16, 17, 24, 30, 37, 38, 87
SMS	7, 9, 48, 49, 56, 71, 86, 87
SMTP	23, 38, 54, 71
SQL database	7, 71
SSI	42, 58, 71
SSL	9, 27, 35, 38, 71, 87
stop bits	47, 66, 68-70
system configuration	15, 16, 29, 38, 43, 54, 58, 59, 88
TCP/IP	6, 7, 9, 51, 54, 55, 86, 87
telephony	56, 71
time server	22
Time settings	22
time stamp	10, 47, 48, 51, 52, 87
time zone	22
timers	8, 48, 53
timeval	52
USB	7, 57
user interface	7-9, 11, 15, 40, 41, 43, 51, 58, 59, 61, 71, 88
user profiles	7, 15, 17, 18, 24, 25, 28-30, 35, 37, 38
Viewpoint	11, 51, 62, 64, 65
web browser	7, 9, 15, 34, 36, 61, 63, 88
web interface	15, 43, 88
web root	35, 36, 58, 75
WebGUI	8, 11, 12, 14, 38, 42, 51, 53, 58, 59, 61-65, 71, 75, 83, 86, 88
WebNotify	38, 42, 59, 60, 71, 86, 88
XML export	71

## Notice to the user

### **IMPORTANT - READ CAREFULLY:**

This CER End-User License Agreement ("EULA") is a legal agreement between you (either an individual or a single entity) and CER International bv for the CER software product FieldCommander, which includes computer software and may include associated media, printed materials, and "online" or electronic documentation ("SOFTWARE"). The SOFTWARE also includes any updates and supplements to the original SOFTWARE provided to you by CER. Any software provided along with the SOFTWARE that is associated with a separate end-user license agreement is licensed to you under the terms of that license agreement. By installing, copying, downloading, accessing, or otherwise using the SOFTWARE, you agree to be bound by the terms of this EULA. If you do not agree to the terms of this EULA, do not install or use the SOFTWARE; you may, however, return it to your place of purchase for a full refund within 10 days of the date you acquired it.

## Software License Agreement

The SOFTWARE is protected by copyright laws and international copyright treaties, as well as other intellectual property laws and treaties. The SOFTWARE is licensed, not sold.

### **GRANT OF LICENSE**

You may install, use, access, display, run, or otherwise interact with ("RUN") one copy of the SOFTWARE, or any prior version for the same operating system, on your FieldCommander. You are obtaining no rights on the SOFTWARE except those given in this limited license.

### **OWNERSHIP**

You may not translate, reverse engineer, decompile, or disassemble the SOFTWARE, except and only to the extent that such activity is expressly permitted by applicable law notwithstanding this limitation. You may not rent, lease, or lend the SOFTWARE. You may not modify the SOFTWARE or merge all or any part of the SOFTWARE in another program. The SOFTWARE is licensed as a single product. Its component parts may not be separated for use on more than one COMPUTER.

### **NO TRANSFER**

You may not sublicense the SOFTWARE. You may not transfer the SOFTWARE to a third party unless you cease all use of it, transfer all copies of it and accompanying Documentation, and the transferee agrees to be bound by the terms of this Agreement.

### **TERM**

This License shall continue for as long as you use the SOFTWARE. However, it will terminate if you fail to comply with any of its term or conditions. Upon such termination you must immediately cease using the SOFTWARE and must follow CER International's instructions regarding return of the Software. ALL DISCLAIMERS HEREIN SHALL SURVIVE TERMINATION.

### **LIMITED WARRANTY**

CER International warrants that (a) the HARDWARE will be free from defects in materials and workmanship under normal use and service for a period of one year from the date of receipt; and (b) the SOFTWARE accompanying the HARDWARE will perform substantially in accordance with the accompanying Product Manual(s) for a period of 90 days from the date of receipt. Any implied warranties on the HARDWARE and SOFTWARE are limited to one (1) year and 90 days, respectively. Some states/jurisdictions do not allow limitations on duration of an implied warranty, so the above limitation may not apply to you.

### **CUSTOMER REMEDIES**

CER International's entire liability and your exclusive remedy shall be, at CER International's option, either (a) return of the price paid or (b) repair or replacement of the HARDWARE or SOFTWARE that does not meet the above Limited Warranty. This Limited Warranty is void if failure of the HARDWARE or SOFTWARE has resulted from accident, abuse, or misapplication. Any replacement HARDWARE or SOFTWARE will be warranted for the remainder of the original warranty period or 30 days, whichever is longer.

### **NO OTHER WARRANTIES**

TO THE MAXIMUM EXTENT PERMITTED BY APPLICABLE LAW, CER INTERNATIONAL DISCLAIMS ALL OTHER WARRANTIES, EITHER EXPRESS OR IMPLIED, INCLUDING BUT NOT LIMITED TO IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE, WITH RESPECT TO THE SOFTWARE, THE ACCOMPANYING PRODUCT MANUAL(S) AND WRITTEN MATERIALS, AND ANY ACCOMPANYING HARDWARE. THE LIMITED WARRANTY CONTAINED HEREIN GIVES YOU SPECIFIC LEGAL RIGHTS. YOU MAY HAVE OTHERS WHICH VARY FROM STATE/JURISDICTION TO STATE/JURISDICTION

**NO LIABILITY FOR CONSEQUENTIAL DAMAGES**

TO THE MAXIMUM EXTENT PERMITTED BY APPLICABLE LAW, CER INTERNATIONAL AND ITS SUPPLIERS SHALL NOT BE LIABLE FOR ANY OTHER DAMAGES WHATSOEVER (INCLUDING, WITHOUT LIMITATION, DAMAGES FOR LOSS OF BUSINESS PROFITS, BUSINESS INTERRUPTION, LOSS OF BUSINESS INFORMATION, OR OTHER PECUNIARY LOSS) ARISING OUT OF THE USE OF OR INABILITY TO USE THIS CER INTERNATIONAL PRODUCT, EVEN IF CER INTERNATIONAL HAS BEEN ADVISED OF THE POSSIBILITY OF SUCH DAMAGES. IN ANY CASE, CER INTERNATIONAL'S ENTIRE LIABILITY UNDER ANY PROVISION OF THIS AGREEMENT SHALL BE LIMITED TO THE AMOUNT ACTUALLY PAID BY YOU FOR THE SOFTWARE.

**GENERAL**

This License is the entire agreement between us, supersedes any other agreement or discussions, oral or written and may not be changed except by a written signed agreement. CER International accepts no liability for any damages whatsoever arising out of the use of or inability to use this software, direct and/or indirect. This License shall be governed by and construed in accordance with the laws of the Netherlands. If any provision of this License is declared by a Court of competent jurisdiction to be invalid, illegal, or unenforceable, such provision shall be severed from the License and the other provisions shall remain in full force and effect. The parties have requested that this Agreement and all documents contemplated hereby be drawn up in English.

**GOVERNMENT RESTRICTED RIGHTS**

The HARDWARE, SOFTWARE and user documentation are provided with RESTRICTED RIGHTS. Use, duplication, or disclosure by the Government is subject to restrictions. Manufacturer is CER International bv, Roosendaal, the Netherlands. Should you have any questions concerning this Agreement, or if you desire to contact CER International for any reason, please write: Customer Service Dept., CER International bv, PO Box 258, NL 4700 AG Roosendaal, The Netherlands.

## Third-party software

This product contains software under the GNU General Public License and other open source licenses. Copies of these licenses and information about obtaining the GPL source code is available on the web at <http://www.cer.com>. If you would like a copy of the GPL source code contained in this product shipped to you on CD for costs only covering preparing and mailing the CD, contact CER International bv, FieldCommander Product Management, PO Box 258, 4700 AG Roosendaal, The Netherlands.

---

Portions copyright © 1995,1998,1999,2000 by Jef Poskanzer <jef@acme.com>. All rights reserved.

THIS SOFTWARE IS PROVIDED BY THE AUTHOR AND CONTRIBUTORS "AS IS" AND ANY EXPRESS OR IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE ARE DISCLAIMED. IN NO EVENT SHALL THE AUTHOR OR CONTRIBUTORS BE LIABLE FOR ANY DIRECT, INDIRECT, INCIDENTAL, SPECIAL, EXEMPLARY, OR CONSEQUENTIAL DAMAGES (INCLUDING, BUT NOT LIMITED TO, PROCUREMENT OF SUBSTITUTE GOODS OR SERVICES; LOSS OF USE, DATA, OR PROFITS; OR BUSINESS INTERRUPTION) HOWEVER CAUSED AND ON ANY THEORY OF LIABILITY, WHETHER IN CONTRACT, STRICT LIABILITY, OR TORT (INCLUDING NEGLIGENCE OR OTHERWISE) ARISING IN ANY WAY OUT OF THE USE OF THIS SOFTWARE, EVEN IF ADVISED OF THE POSSIBILITY OF SUCH DAMAGE.

---

Portions copyright © 1996 - 2001, Daniel Stenberg, <daniel@haxx.se>. All rights reserved.

Permission is hereby granted, free of charge, to any person obtaining a copy of this software and associated documentation files (the "Software"), to deal in the Software without restriction, including without limitation the rights to use, copy, modify, merge, publish, distribute, and/or sell copies of the Software, and to permit persons to whom the Software is furnished to do so, provided that the above copyright notice(s) and this permission notice appear in all copies of the Software.

THE SOFTWARE IS PROVIDED "AS IS", WITHOUT WARRANTY OF ANY KIND, EXPRESS OR IMPLIED, INCLUDING BUT NOT LIMITED TO THE WARRANTIES OF MERCHANTABILITY, FITNESS FOR A PARTICULAR PURPOSE AND NONINFRINGEMENT OF THIRD PARTY RIGHTS. IN NO EVENT SHALL THE COPYRIGHT HOLDER OR HOLDERS INCLUDED IN THIS NOTICE BE LIABLE FOR ANY CLAIM, OR ANY SPECIAL INDIRECT OR CONSEQUENTIAL DAMAGES, OR ANY DAMAGES WHATSOEVER RESULTING FROM LOSS OF USE, DATA OR PROFITS, WHETHER IN AN ACTION OF CONTRACT, NEGLIGENCE OR OTHER TORTIOUS ACTION, ARISING OUT OF OR IN CONNECTION WITH THE USE OR PERFORMANCE OF THIS SOFTWARE.

## Trademarks

CER and FieldCommander are registered trademarks of CER International bv.

All other product names and services identified throughout this book are trademarks or registered trademarks of their respective companies. They are used throughout this manual in editorial fashion only and for the benefit of such companies. No such uses, or the use of any trade name, is intended to convey endorsement or other affiliation with this manual.

## Copyrights

Copyright © 2007 CER International bv. All rights reserved.

No part of this publication may be reproduced in any form, or stored in a database or retrieval system, or transmitted or distributed in any form by any means, electronic, mechanical photocopying, recording, or otherwise, without the prior written permission of CER International bv, except as permitted by the Copyright Act of 1976 and except that program listings may be entered, stored and executed in a computer system.

THE INFORMATION AND MATERIAL CONTAINED IN THIS MANUAL ARE PROVIDED "AS IS," WITHOUT WARRANTY OF ANY KIND, EXPRESS OR IMPLIED, INCLUDING WITHOUT LIMITATION ANY WARRANTY CONCERNING THE ACCURACY, ADEQUACY, OR COMPLETENESS OF SUCH INFORMATION OR MATERIAL OR THE RESULT TO BE OBTAINED FROM USING SUCH INFORMATION OR MATERIAL, NEITHER CER INTERNATIONAL BV NOR THE AUTHORS SHALL BE RESPONSIBLE FOR ANY CLAIMS ATTRIBUTABLE TO ERRORS, OMISSIONS, OR OTHER INACCURACIES IN THE INFORMATION OR MATERIAL CONTAINED IN THIS MANUAL, AND IN NO EVENT SHALL CER INTERNATIONAL BV OR THE AUTHORS BE LIABLE FOR DIRECT, INDIRECT, SPECIAL, INCIDENTAL, OR CONSEQUENTIAL DAMAGES ARISING OUT OF THE USE OF SUCH INFORMATION OR MATERIAL.

from Device to Enterprise

**CER**<sup>®</sup>

The latest documentation is available  
from <http://www.cer.com>

FieldCommander is a product of:

CER International bv  
Postbus 258  
NL 4700 AG Roosendaal  
The Netherlands  
TEL: +31 (0)165-557417  
FAX: +31 (0)165-562151  
[www.cer.com](http://www.cer.com)

Part no. FCSWUG, revision 3.5.20