WebGate

User manual

PRELIMINARY VERSION

LEGGI E CONSERVA QUESTE ISTRUZIONI

READ AND SAVE THESE INSTRUCTIONS

CAREL

Technology & Evolution

Contents

Introduction

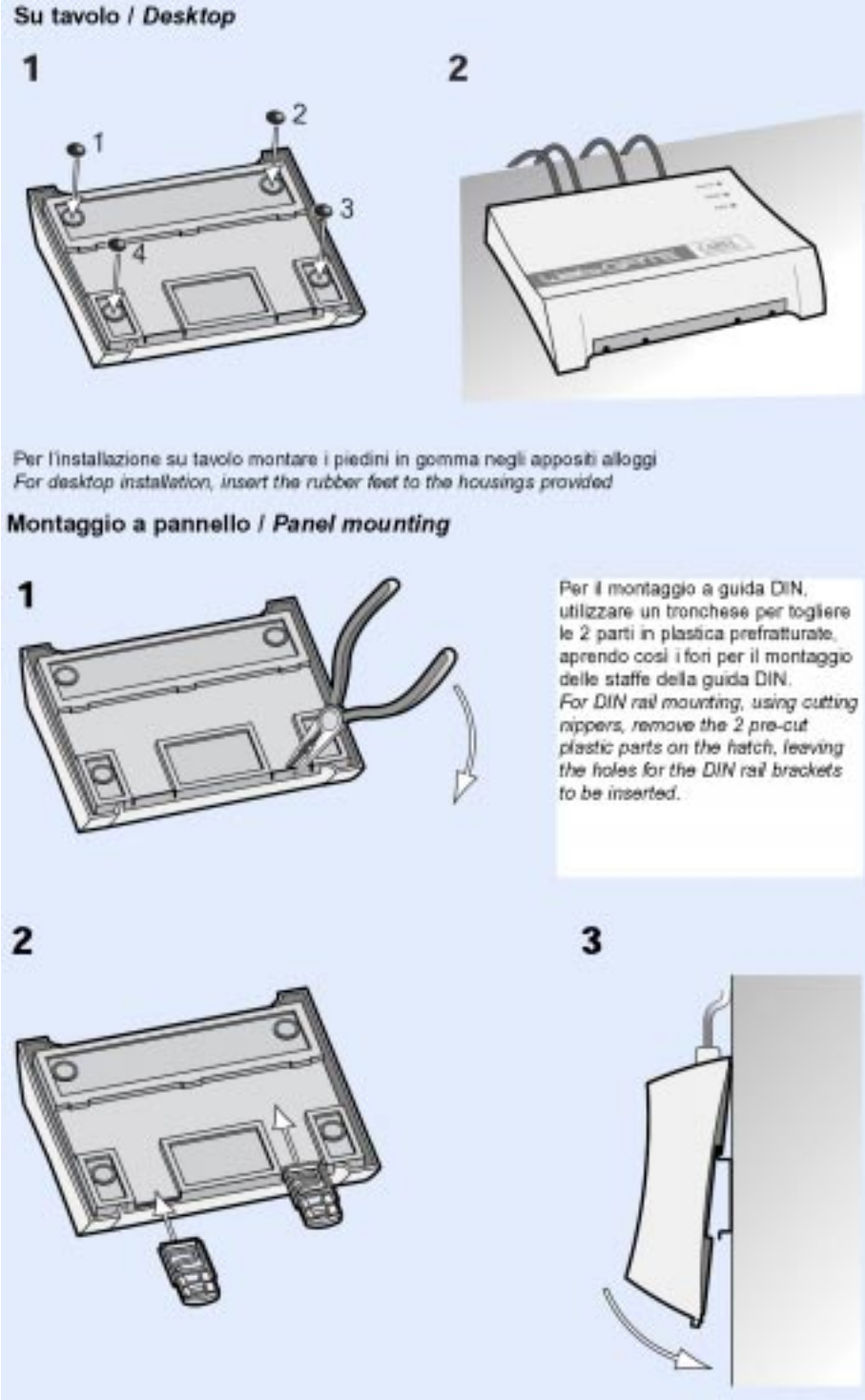
WebGate is a innovative electronic device that measures the same size as a normal desktop modem. It features avant-garde technology for connecting all Carel controllers to the local network based on the standard Ethernet™ and TCP/IP.
Ethernet™ is a widespread, fast, economical and reliable communication standard, and is the physical support for the TCP/IP protocol. Ethernet™ networks are now used in numerous different types of systems, and can thus be exploited as the backbone for the transmission of data, without needing to add further wiring in the supervision of the instruments.
Ethernet™ and TCP/IP are the technologies that underlie the Internet, and as a result WebGate allows the use of tools, such as web browsers, for performing diagnostic functions and the local and remote monitoring of the systems.
The TCP/IP protocol can also be used as the support for other protocols, when transferring the data from the controls connected in the Ethernet™ network to a local or remote supervisor.
WebGate is easy to configure either via the web or using an RS232 serial connection: as a matter of fact, only a minimum configuration is sufficient for setting its individual IP address.

WEB SERVER functions
One of the main functions performed by WebGate is the WebServer function: using the HTTP (Hyper Text Transfer Protocol) standard, WebGate can "serve" web pages to client computers connected in a local network or WAN.
The web pages are written based on the characteristics of the specific installation, for the management of the data from the instruments via the Carel RS485 network. The user can thus display and modify the installation parameters using an Internet browser, such as Microsoft® Internet Explorer™ or NetScape Navigator®, as the interface, and typing in the IP address of the WebGate.
The pages can be written using numerous readily available HTML editors that can be used to create even very complex web pages with only basic knowledge of the web page programming language.

FTP protocol
The web pages relating to the specific installation are saved to the "flash" memory inside the WebGate via FTP (File Transfer Protocol), the protocol used to transfer files on TCP/IP networks. Then, using simple "drag and drop" operations with the mouse, the web page HTML files can be copied from the user's computer to the WebGate.

SNMP protocol
WebGate is also an SNMP gateway. It in fact converts the Carel communication protocol to the Simple Network Management Protocol (SNMP), the protocol used to send data from the instruments on the Ethernet™ – TCP/IP network to a local or remote supervisor for subsequent processing.
SNMP is a protocol developed specifically for the management of TCP/IP networks, founded in 1988 based on the specifications of the IAB (Internet Administration Board), the body that supervises the Internet protocol. This is thus a protocol developed specifically for the management of data on TCP/IP networks, and is consequently very widespread and suitable for the specific application.
The SNMP functions are complementary to the HTTP functions: as a result, an Internet browser, which uses HTTP, can be used to access the installation data for modification or monitoring. Nonetheless, it is not possible to perform the other typical supervisor functions, such as logging the data, managing alarms, etc. These functions in fact require a supervisor that is continuously connected to the WebGate, and which receives the data from the controllers via the TCP/IP network. These are then sent to the WebGate via the SNMP protocol and can be managed by a supervisor with SNMP management functions.

1. Installation



Installation Warnings

Avoid installing the boards in environments with the following characteristics:

- relative humidity above 90%;
- strong vibrations or knocks;
- exposure to jets of water;
- exposure to aggressive and polluting agents (e.g.: sulphurous and ammonia gases, saline mists, smoke) which may cause corrosion and/or oxidation;
- high levels of magnetic and/or radio-frequency interference (thus avoid installation near transmitting antennas);
- exposure of the device to direct sunlight and atmospheric agents in general;
- large and rapid fluctuations in ambient temperature;
- environments where explosives or mixes of inflammable gases are present;
- exposure to dust (formation of corrosive patina with possible oxidation and reduction
- of insulation).

1.1 Connections

- **WebGate connections are accessible from the back panel of the unit, as indicated below:**

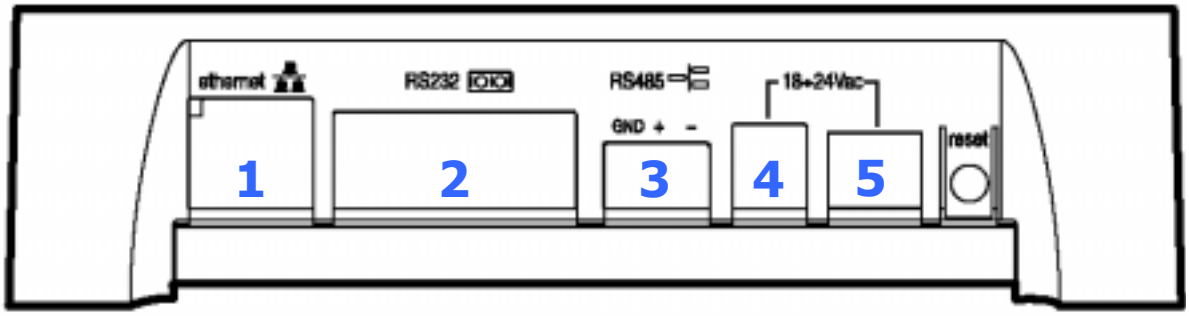


Fig. 1.1.1

1. **Ethernet 10BaseT connector** to corporate LAN.
2. **RS232 DTE interface** to connect a local console
3. **RS485 interface to Carel Network**, used to connect the Carel devices
4. **Plug connector for the 18V$_{AC}$ power adapter** (desktop installation)
5. **Connector for the 24V$_{AC}$ power supply** (panel mounting)

When connecting WebGate the following WARNINGS should be heeded:

1. Voltages different from the power ratings will seriously damage the system.
2. Use cable-ends which are suitable for the terminals being used. Loosen each screw and insert the cable-end, then tighten the screws. On completing the operation lightly tug the cables to check that they are correctly inserted.
3. Avoid touching or nearly-touching electronic components mounted on the boards to avoid electrostatic discharges (extremely damaging) from the operator to the components.
4. Separate as much as possible the signal cables from any power cables to avoid possible electromagnetic influence. Never insert power cables (including mains cables) and RS485, RS232 or Ethernet cables in the same channels.
5. Never try to take the unit apart or modify it in any way. Doing so creates the danger of fire and electric shock.

1.1.1 Connection of the Ethernet cable

WebGate uses a RJ45, 10BaseT connector, 10Mb/s interface. When connecting to a hub or switch use a straight cable patch. When connecting directly to a PC use a cross cable instead.

1.1.2 Connection of the RS232 interface

WebGate is provided with a standard DTE interface with a 9 pin male DB-9 connector: In the following table are depicted the standard RS232 signals:

Pin N°	Abbreviation	Description	Direction
1	CD	CARRIER DETECT	From DCE
2	RD	RECEIVE DATA	From DCE
3	TD	TRANSMIT DATA	To DCE
4	DTR	DATA TERMINAL READY	To DCE
5	SG	SIGNAL GROUND	---
6	DSR	DATA SET READY[1]	From DCE
7	RTS	REQUEST TO SEND	To DCE
8	CTS	CLEAR TO SEND[1]	From DCE
9	RI	RING INDICATOR[2]	From DCE

Tab. 1.1.2.1

[1]These pins may be unconnected in some WebGate models
[2]This pin is unconnected in all WebGate models.

To connect WebGate to a PC use a shielded null-modem cable. Only TD, RD and SG lines are strictly required. Remember that since the PC and WebGate are provided with a DTE interface, TD and RD lines must be swapped:

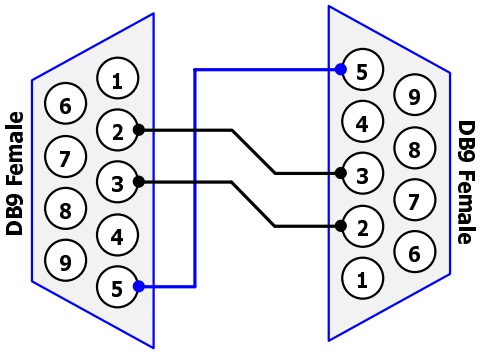


Fig. 1.1.2.1

1.1.3 Connection of the RS485 interface

The wire for the connection of WebGate with Carel peripherals in RS485 suggested by Carel is:

- 2 twisted wires,
- shielded, preferably with a continuity wire,
- section AWG20 (0,5mm^2) or AWG22 (0,32÷0,38mm^2),
- wire capacity lower than 100pF/m

(the models 8761 and 8762 of *Belden*, for example, satisfy the previous requirements).

Always connect the supplied 120Ω terminator resistor to the remote end of the network between the "Tx/Rx+" and "Tx/Rx-" wires.

Example:

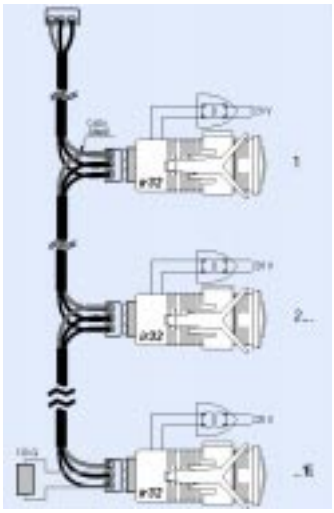


Fig. 1.1.3.1

1.1.4 Connection of the Power Supply

Power supply connectors 4 and 5 (Fig. 1.1.1) are electrically equivalent. Do not use the two power supply connectors at the same time. Connector 4 is for desktop installation. Use ONLY the power adapter supplied on request by Carel (code TRA1806ITA). The use of different power adapters may damage the hardware.
Connector 5 is for panel mounting. Use a safety transformer rated to at least 6VA. The use of the power adapter supplied on request by Carel is suggested (code TRA1810DIN). It is obligatory to insert in series with the unit power supply a 500mAT fuse.

2. User Interface

All WebGate functions can be controlled using HTML pages or RS232 console interface. However, three multicolor LEDs and a "reset" button are provided to simplify installation.

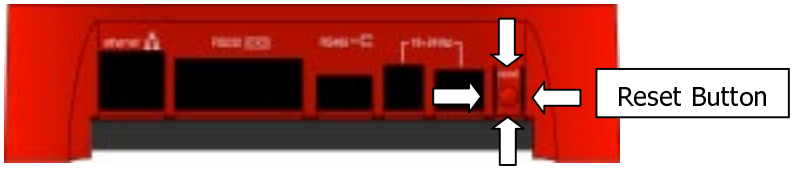


Fig 2.1: back side of WebGate

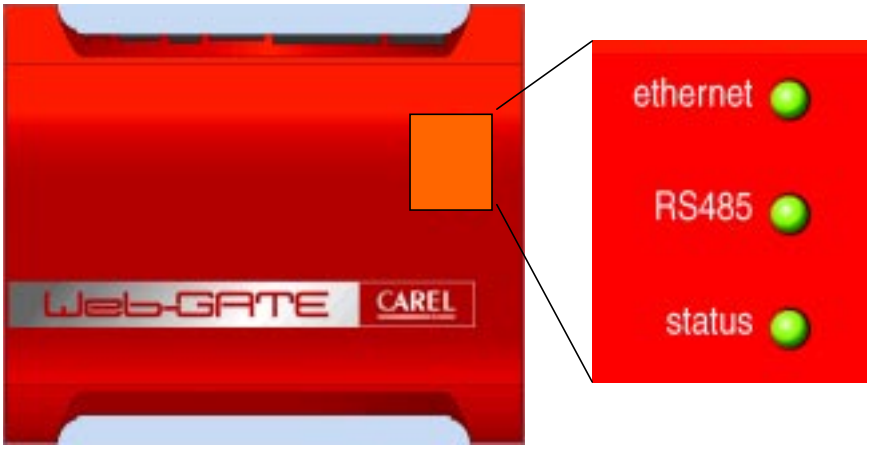


Fig 2.2: top side of WebGate

2.1 LED

2.1.1 Ethernet

The top LED gives information about the Ethernet link and connection.

Normal behaviour
The normal LED behaviour is indicated in the table below:

Color	Meaning
Off	Connected (Link), no data transfer in progress.
Green (flashing)	Link, data transfer in progress.
Yellow	No link, WebGate is trying a transmission.
Red	No link. Generally this indicates disconnected cable, wrong cabling or remote interface (generally a PC or a hub) powered off.

Tab. 2.1.1.1

Power up:
When WebGate is powered up, the Ethernet LED becomes red for some seconds until the interface is properly initialized.

2.1.2 RS485

The central LED displays the status of the RS485 Carel network.

Normal behaviour
This LED normally visualizes at intervals of 500ms the status of each unit that should be connected to the RS485 interface. In this manner, it is possible to evaluate quickly the network status.

In the following table is indicated the normal color meanings of the LED:

Color	Meaning
Off	Unit not scanned yet.
Green	Unit online (data is received from the unit)
Yellow	"Heartbeat", used to display that the network scan is active (WebGate is trying to transmit to units).
Red	• **Unit offline**

• Tab. 2.1.2.1

Note: peripherals are scanned from address "1" up to the address indicated from the function "`MaxDevs`" (see **WebGate Script Functions**).

An example sequencing of colors is reported in the example figure below; the following assumptions are made:

- 3 units connectable (function "`MaxDevs`" is set to "3" devices)
- units 1 and 2 are online,
- unit 3 is disconnected.

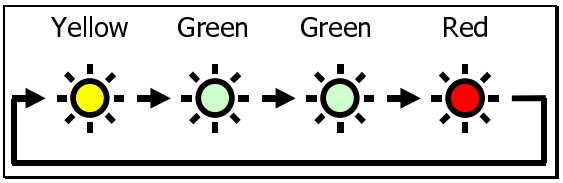


Fig. 2.1.2.1

Network cable disconnected
If the network cable is disconnected, after some seconds the LED will light up red and yellow only.

Power up:
When WebGate is powered up, the LED doesn't light up for some seconds until the interface is properly initialized. After initialization, please wait some seconds to get all the units online.
If the LED remains off (or if it blinks "red" sometimes) during normal operation, you can use the "`Status485`" function from the console to obtain further information (see **WebGate Script Functions**).

2.1.3 Status

The bottom LED displays the generic status of the WebGate, as indicated in the following table:

Color	Meaning
Off	Power off
Green	Normal operation
Yellow	WebGate is busy
Red	Initialization at power-up (about one second) or fatal error

Tab. 2.1.3.1

Power-up or Reboot:

- **When WebGate is powered up or restarted with the "`Reboot`" command, the Status LED becomes yellow for few seconds until the interface is properly initialized.**

Abnormal operation:
If during power-up the Status LED continuously blinks red or don't light up at all, a serious error is occurred.

Firmware Update Procedure:
The firmware update is a procedure that starts after a reboot and that may take up some minutes. During this time the status LED will continuously light yellow.

Important WARNINGS.
Never remove power supply when WebGate is busy (status LED is yellow), since this may cause loss of the configuration and user files.

Absolutely never remove power supply when WebGate is performing a firmware update (status led is yellow), since this may severely damage the device and require technical assistance.

2.2 Reset Button

The reset button is useful to restore the WebGate configuration to its factory default. This may be necessary in particular if the settings of the communication parameters were changed and forgotten.

Two reset modes are provided: Configuration Reset and Total Erase.

2.2.1 Configuration Reset

To restore basic configuration settings to their original values press the reset button and hold it down for **2 seconds**.
When the button is released, the status LED will blink alternatively yellow and green for some seconds. When the LED will return back to a steady green the following parameters will gain their default values:

Parameter	Default Values
IPAddress	192.168.0.250
NetMask	0.0.0.0
Gateway	255.255.255.255 (Disabled)
Baudrate485	19200
MaxDevs	16
Baudrate232	19200
TRAPIPAddress	255.255.255.255 (Disabled)

Tab. 2.2.1.1

2.2.2 Total Erase:

This option is provided basically as a “last chance” aid when passwords are lost.

Important WARNING: **Please note that as a security measure, all user files will be erased from this procedure (included the customized web pages), and the file system will return to its factory (or last update) state.**

To perform a total erase press the reset button and hold it down for **10 seconds**.
When the button is released, the status LED will blink alternatively yellow and red for some seconds. When the LED will return green again, in addition to the default values indicated in the table above (Tab. 2.2.1.1),, the following changes will also occur:

- All user files will be erased (with any access level : “guest”, “user”, “supervisor” or “administrator”)
- The user table will be erased, removing any user name and relevant passwords.

The following *items* are not changed instead:

- Factory files or files updated with a distribution upgrade
- The parameters in the table below:

Parameters NOT changed by reset button
RWCommunity
SNMPSysname
SNMPSyscont
SNMPSysloc
TRAPCommunity
ROCommunity

Tab. 2.2.2.1

3. HTML Configuration Interface

WebGate is shipped from the factory with some predefined pages provided to configure easily the unit.
As indicated in the following paragraph, every information available from these pages is obtained through the use of a script function: to obtain more information about any function see **WebGate Script Functions**.
In addition, WebGate users can create custom pages with the same capabilities as the original shipped from factory (see **Creating a custom Web page on the WebGate**).

The page is best viewed with a browser enabled for JavaScript and Cascading Style Sheets (CSS) (for example, Internet Explorer™ 5.0) on a monitor displaying a resolution of at least 800x600 pixels, 32bpp.

3.1 Information page

The *Information* page is the first page displayed when you connect to WebGate.
This is the only page accessible to everybody. To access to any other configuration page is required an *administrator* access level instead.

Fig. 3.1.1

The visible fields are explained below:

Field	Description	Function
Firmware Version	Latest WebGate software update version (revision)	**SWVersion**
Hardware Tracking	WebGate hardware version (revision). The value indicated here is only indicative. When contacting for Carel support, always report the release and serial number indicated on the label	**HWVersion**
Files Space	Indicates how many bytes and file locations are free to store files in the WebGate file system.	**FreeFiles**
Last Update Status	Return the possible last error occurred (if any) when the last reboot was performed and a firmware update was tried. Please note that the indication “Update file not found” is normal and it not indicates an error.	**UpdateStatus**

Tab. 3.1.1

3.2 Configuration pages

To access to any configuration page is required an *administrator* access level.

3.2.1 *General* configuration tab

The *General* configuration tab is the first page visible when you click on the “Configure” link on the left navigation panel:

Fig. 3.2.1.1

The visible fields are explained below:

Field	Description	Function
REBOOT WEBGATE	This checkbox is used to restart the WebGate in a way similar to the power-up. This is used mainly when you want to proceed for a firmware update. To reboot WebGate, simply check the box and press the "Apply" button. Please note that the operation may take some minutes when a software update is performed and, in addition, you will have to manually refresh the page pressing the "update" button of the browser.	**Reboot**

Tab. 3.2.1.1

3.2.2 *Network* configuration tab

The *Network* tab is used to configure Ethernet interface:

Fig. 3.2.2.1

The visible fields are explained below:

Field	Description	Function
IP Address	IP Address of WebGate itself. By default, this value is set to "192.168.0.250".	**Reboot**
Subnet Mask	Network Mask Pattern. If you don't need to use a gateway you can safety set this value to its default of 0.0.0.0	**NetMask**
Gateway	Network Gateway Address. If you don't need to use a gateway set this value to its default of 255.255.255.255	**Gateway**
MAC Address	Ethernet hardware address. This value can be useful to system administrators and cannot be changed.	**MACAddress**
Network Statistics	This is a list of network statistics. This may be useful to system administrators. For a detailed description please refer to the "`NetStat`" command paragraph.	**NetStat**

Tab. 3.2.2.1

WARNING.
If you aren't accustomed with network management we suggested to consult the chapter **WebGate Script Functions** to better understand how the related functions works. In such a situation, to obtain a valid IP Address, Subnet Mask and Gateway address contact your system administrator.

3.2.3 *RS845* configuration tab

The *RS485* tab is used to configure the RS485 interface towards Carel Network:

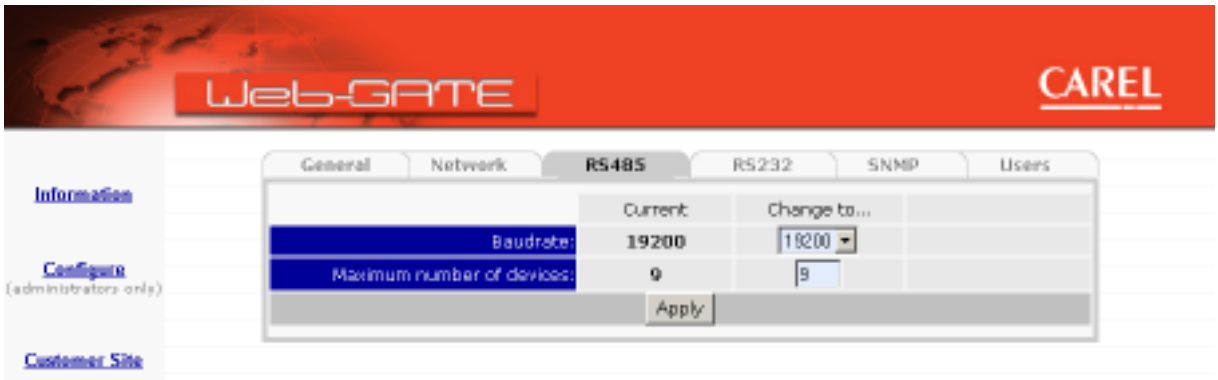


Fig. 3.2.3.1

The visible fields are explained below:

Field	Description	Function
Baudrate	Baudrate for the RS485 interface	**Baudrate485**
Maximum number of devices	Maximum number of devices connected to WebGate. Is suggested to set this value to the number of actually connected units to allow a faster response and a proper "RS485" LED working.	**MaxDevs**

Tab. 3.2.3.1

3.2.4 *RS232* configuration tab

The *RS232* tab is used to configure the RS232 interface for the serial console interface:

Fig. 3.2.4.1

The visible fields are explained below:

Field	Description	Function
Baudrate	Baudrate for the RS232 interface	**Baudrate232**

Tab. 3.2.4.1

3.2.5 *SNMP* configuration tab

The *SNMP* tab is used to configure the SNMP protocol features:

Fig. 3.2.5.1

The visible fields are explained below:

Field	Description	Function
Read Only Community Name	SNMP Read Only Community Name	**ROCommunity**
Read/Write Community Name	SNMP Read/Write Community Name	**RWCommunity**
System Name	SNMP System Name	**SNMPSysname**
System Contact	SNMP Contact Name (Operator)	**SNMPSyscont**
System Location	SNMP System Location	**SNMPSysloc**
NSM Trap IP Address	Trap Manager IP Address (default not used: 255.255.255.255)	**TrapIPAddress**
Trap Community Name	SNMP Trap Community Name	• TrapCommunity
Agent Release	WebGate Agent software release	**SNMPAgentRel**

Tab. 3.2.5.1

WARNING
The maximum length of each string SNMP is:

Max. length of the system variables in the WebGate	
System Variable	**Max. Length**
SysName	39 characters
sysContact, sysLocation	79 characters

Tab. 8.7.1

Moreover, the string have to be composed *only* of alphanumeric characters (0…9, A…Z and a…z). No spaces or punctuation marks are allowed.

For a complete description on SNMP please see **WebGate SNMP Protocol**.

3.2.6 *Users* configuration tab

The *Users* tab is used to configure the WebGate Users:

Fig. 3.2.6.1

A complete description on how to configure users can be found in **User Management.**

3.3 Customer Site link

The "*Customer Site*" link provides a fixed link to the "/default.html" page that should be your main customized page.

4. Interface RS 232 (Console)

4.1 Introduction

The console user interface RS232 is a fast and convenient way to configure WebGate and to experiment with its functions.

It is basically a command line interface accessed via the RS232 port using a PC and a terminal emulation program.
Since WebGate is provided with a DTE interface connector, it must be connected to the PC with a *null-modem* cable. Only the TX and RX signals are required.

4.2 Settings

The following settings are required when configuring the terminal emulator:

- Select the communication port where the WebGate is connected to (generally COM1 or COM2).
- Set the serial port configuration to **8 bit data**, **no parity**, **1 stop bit**, **no flow control**
- Make sure that the terminal is configured at the same communication speed of the WebGate speed. The default serial port baudrate of WebGate is **19200 baud**. This value can be changed using the HTML interface (see ***RS232* configuration tab**). However, make sure that the terminal is configured in the same way

The following settings are suggested when configuring the terminal emulator:

- Use the TTY emulation mode
- Disable local characters echo. Characters are remotely echoed from WebGate.
- enable the "delete with backspace" option when available, to allow the correct handling of the "backspace" character sent by WebGate
- Don't add a "LF" after transmitted lines
- Don't' add a "LF" after received lines (this is performed automatically by WebGate).

4.3 A step-by-step configuration example

In this paragraph is described step-by-step how a terminal emulator can be used as console interface for the WebGate.
For example we will use Hyperterminal®, the terminal emulator supplied with Microsoft® Windows©. However, any terminal emulator, if properly configured, may be used.

1. Select and run *Hypertrm.exe* from **Start>Programs>Accessories>Hyperterminal**; the following window will appear (it may be differ somewhat depending on your operative system language):

Fig. 4.3.1

2. Choose an icon and a name for the console section you want, for example WebGate and click OK. The following window appears:

Fig. 4.3.2

3. Select the COM[x] serial port for the connection between PC and WebGate, i.e. COM1, and click the OK button, so you can choose now the communication parameters from the following window:

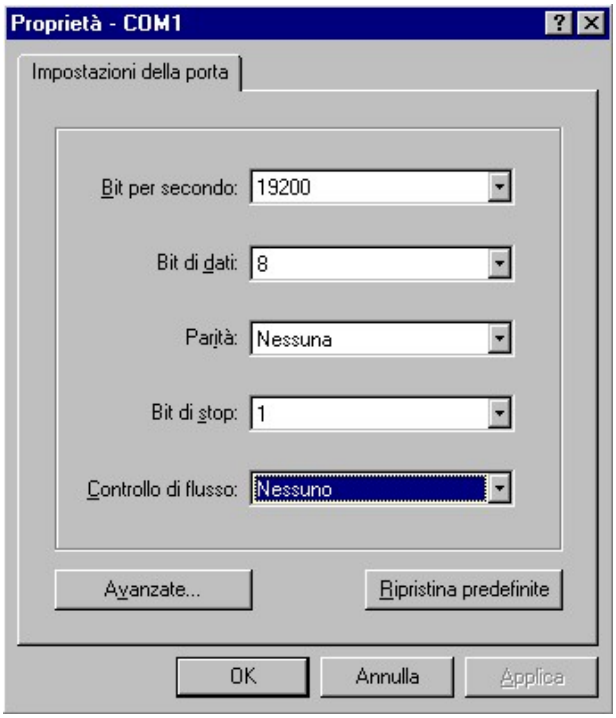


Fig. 4.3.3

4. Select: **19200 baud**, **8 bit data**, **no parity**, **1 stop bit**, **no flow control** and click OK.

Now you have the console window for the communication between PC and WebGate working at 19200 bps, which is the default baudrate for the RS232 WebGate serial port.

WARNING: if the WebGate's settings are not the default one and its RS232 baudrate doesn't match the HyperTerminal baudrate, your console will not work correctly. Trying to send one of the available commands, the "help" command for example, the console window will show anomalous characters as in the following figure:

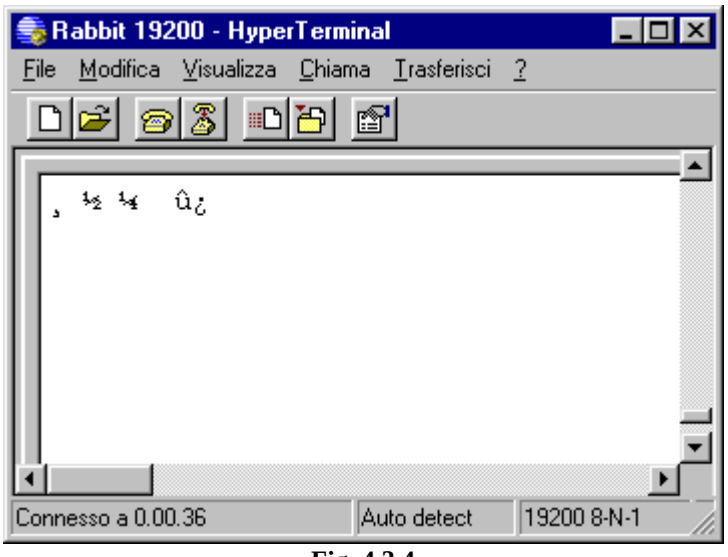


Fig. 4.3.4

If you don't remember the baudrate previously setted to the WebGate's RS232 serial port, the problem can be solved resetting the gateway in order to certainly set its baudrate to 19200 (see **Reset Button**).

4.4 Additional notes about the console

4.4.1 Prompt

When WebGate is ready for a user command, it displays a line beginnig with a “>” sign.

4.4.2 Login

By default, console access is restricted to the same rights of the “anonymous” user.
Since of this, if you are using the *user management* capabilities (see **User Management**) some functions may be restricted to you. To gain full access to console functions, please use the “Login” command (see **WebGate Script Functions**).
When WebGate is shipped from factory, the anonymous user have “administrator” rights. Since of this, for the first configuration you don’t need to use the Login command.

4.4.3 Limitations using expressions

When using a function with the terminal, please remember that expressions following the assignment operator “=” must not contain spacing characters:

```
set(test) = test+ 1     ← Wrong
set(test) = 3 + 1       ← Wrong
set(test) = test+1      ← OK
```

4.4.4 Delays after function execution

- After the execution of some functions (e.g. “`IPAddress`”) a noticeable terminal slow-down may occur and some keyboards characters could be ignored.
- This is a normal and unavoidable behaviour that occurs when the internal configuration file is modified. During this time the status LED will light on yellow to indicate a busy status.

5. File system

WebGate is based on an embedded file system accessible to the user.
File system is used to store HTML pages and any other file that may be useful to the user.
The user can store up to 100 user files in the 400 KBytes storage space reserved to this purpose.
In addition, WebGate comes with some factory files (configuration pages, images, etc.) that are redundant and write protected for increased reliability.

5.1 Files

User files have the following characteristics:

- ✓ Maximum file name length is 127 ASCII characters.
- ✓ The following symbols are ***allowed***:
  A...Z , **a...z** , **0...9** and **%** **+** **-** **.** **,** **;** **=** **@** **_**
  Neither characters with ASCII code less of "33" or greater than 127, or characters listed below are ***not allowed***:
  **** **/** **:** **<** **>** **'** **"** ***** **{** **}** **^** **!** **[** **]** **#** **|** **&** **(** **)** **$** **?** **~** [space]
  For instance, please note that "space" characters inside a name are not allowed.
- ✓ Capital and non-capital letters are preserved, but file search is not case sensitive (in a similar way to MS Windows ™).
- ✓ Files are stored with a "header" of about 150 bytes in sectors of 1KByte each. Consequently, every file will use a memory area a little bit greater than the size of the file itself.
- ✓ Files are characterized from a minimum read access level and a minimum write access level. Next paragraphs describe how this levels can be set for any file.
- ✓ Since WebGate don't have an on board real time clock, every file is stored with a fixed "fake" date of creation (day 1, month 1 of 2002 at 00:00)

5.2 Directories and "Read Access" file protection

The structure of the folders in WebGate is fixed. The users cannot create or cancel the folders.
If it is not necessary to access the user management functions, it is possible to memorize all the files in the main folder. Other wise, it is possible to use the other folders "\user\", "\supervisor\" and "\administrator\".

- **Basically, *the directory tree is used to assign a read access level to a file*:**
  - Files placed in "\" (root) are readable from everybody with "`guest`" access level.
  - Files placed in "\user\" are readable with "`user`" or greater access level.
  - Files placed in "\supervisor\" are readable with "`supervisor`" or greater access level.
  - Files placed in "\administrator\" are readable with "`administrator`" or greater access level.

- When using FTP (see File Transfer Protocol (FTP)), an operator can only see the directory folders he can access to. For example, if the operator has a "`supervisor`" access level, the "\administrator\" directory is hidden.

5.3 "Write Access" file protection

- Every file stored into WebGate is provided with a "write" access protection, distinguished from the "read" access protection, to avoid involuntary or malicious file deletion or modification by unauthorized users.
- Write Access file protection is obtained through FTP and is described in *File Transfer Protocol*.

5.4 Additional Note

- When reading and writing files, the "busy" LED will light up yellow.

6. File Transfer Protocol (FTP)

*Note: before reading the following chapter you must be accustomed with the WebGate File System, described in **File System**.*

File system read access is provided through HTTP. However FTP is required to store files into WebGate.
FTP is a user friendly way to deal with files. Modern FTP clients fully integrates with windows based operating systems, greatly simplifying file transfers. Moreover they provide an extensive error description when something doesn't work properly.

6.1 The FTP client

We suggest to use the freeware program SmartFTP™ (http://www.smartftp.com).
However, nearly any FTP client can be used, but some one is discouraged (for example, Microsoft© Internet Explorer®).
FTP clients generally needs a little configuration to communicate with WebGate, or nothing at all.
However, the following rules must be kept in mind:

- The client must **not** be set to "*Passive Mode*" (PASV)
- The maximum number of threads to use for download must be set to **1**.
- If you are passing through a Firewall or a Proxy Server, make sure FTP protocol is not blocked, since this is a common "security" limit imposed by the firewalls' default settings. If you are having troubles with the connection contact your system administrator.

6.2 "Write Access" file protection

- Every file stored into WebGate is provided with a "write" access protection, distinguished from the "read" access protection, to avoid involuntary or malicious file deletion or modification by unauthorized users.
- **Write Access file protection acts denying "write" or "delete" capabilities through FTP.**

File *write protection* is quite straightforward: every file you upload to the WebGate will get a "write access level" same as your current FTP login access level used for the connection. Only users with an access level equal or greater than this will be allowed to erase or overwrite it.
For example, if you logged in with the FTP client using a "`supervisor`" access level, only supervisors and administrators will be able to change the files you uploaded during this section.
For instance please note that, for the same file, write access protection can be "stronger" than read access protection: for example, if you logged in with the FTP client using a "`supervisor`" access level and you stored a file into the *root* directory, everybody will be able to read the file, but only supervisors and administrators will be allowed to change that file.

6.3 Additional Note

Do not upload to the WebGate a file if the name contains spaces. If the original file name in the host PC contains a space, the name is truncated when copied into WebGate or an error is returned.

6.4 Example

The examples of this chapter were realized using the freeware program SmartFTP® (http://www.smartftp.com).
Below is depicted an example of what you see if you have "`administrator`" access-level.
As a matter of fact, visible folders depend on the access level of the user who is connecting to the WebGate. As described in **User Management** you will be able to see only the folders with an access level equal or lower than yours.
If you have the lowest access level (`guest`) you will be able to see only files loaded in the root folder, since they aren't provided with any read protection.

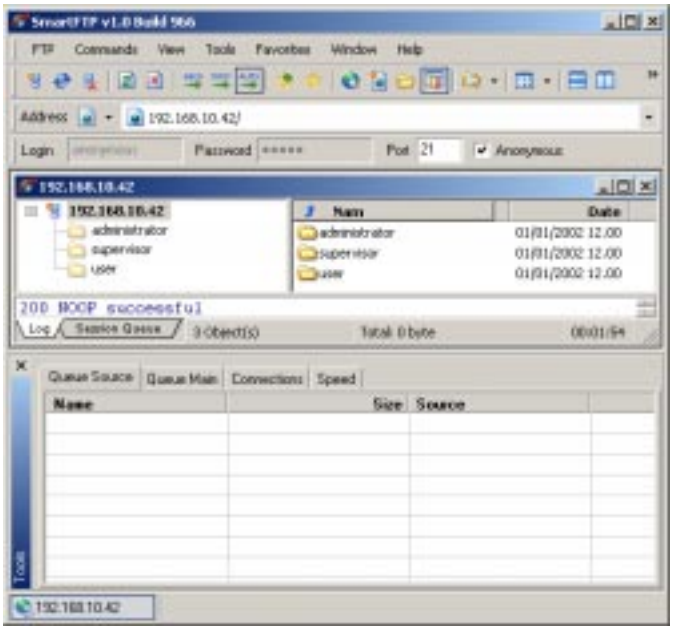


Fig. 6.4.1

If you want to upload a file to the WebGate (pwd.html in the example), giving it "`supervisor`" access-level, you have to load it in the "`supervisor`" folder. Only a user with "`supervisor`" or greater membership will then be able to view such a file.
To upload the file you have to open the "supervisor" folder clicking on it, and using your mouse drag the file from your file manager to the FTP client window, as shown below in Fig. 6.4.2 and 6.4.3.

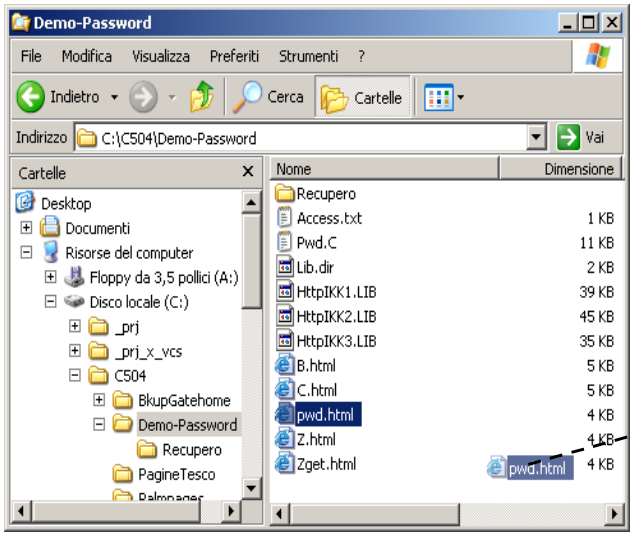


Fig. 6.4.2

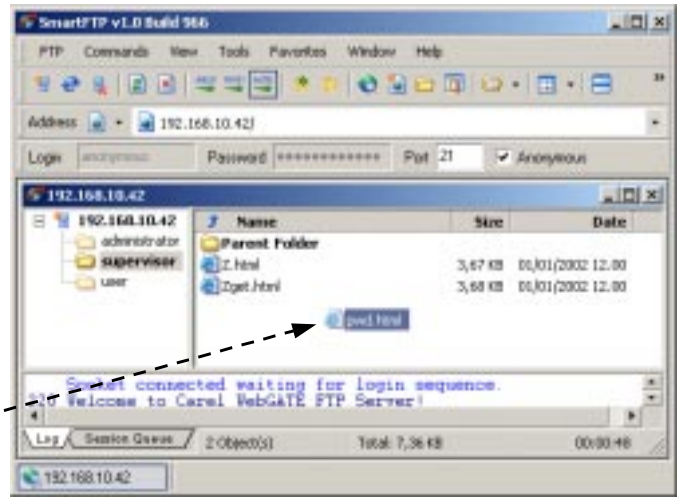


Fig. 6.4.3

7. Creating a custom Web page on the WebGate

7.1 Introduction

This chapter describes how to create a simple Web page which allows you to monitor and control the values of network variables over a local network (LAN) or wide area network (WAN).
It is assumed you have a basic understanding of the HyperText Markup Language (HTML).
In order to serve Web pages that display the network variables to a standard Web browser, you must insert some special "tags" in the HyperText Markup Language of the Web page.
Before returning a Web page to the browser, the Web server embedded in the WebGate parses the page searching for a special HTML tag indicating a network variable or a configuration parameter reference. The Web server, when returning information to the browser, replaces so the current value of the network variable for this tag.
Web pages may be constructed with any off-the-shelf text or HTML editor. All the parameters of all Carel controllers in RS485 can be monitored and modified through Web pages.

7.2 Requirements

You will need the following software:

- A standard FTP client application such as SmartFTP (www.smartftp.com) for uploading your Web pages into the WebGate
- An HTML editor, such as Macromedia® DreamWeaver™ (www.dreamweaver.com).
- Microsoft® Internet Explorer™ (version 5 or higher) or Netscape Communicator™ (version 6 or higher).

You will need also a basic understanding of the HyperText Markup Language (HTML).

7.3 Creating Web Pages

HTML files for the WebGate can be created with any standard text or HTML editor. The WebGate embedded Web server supports standard HTML for defining the structure and format of your Web page, as well as some special HTML tags for retrieving dynamic data elements and processing HTML forms.
HTML and graphic files reside in some special directories of the WebGate flash memory. Approximately 400 KB of space is available for user files (see **File system**).
Files are read and written to the WebGate flash memory using the standard FTP protocol over IP connection (see **File Transfer Protocol (FTP)**).
To retrieve your Web page enter http://192.168.0.250/mywebpage.htm (where 192.168.0.250 is the WebGate's IP address) in the browser's URL window. Be sure to use the proper case; file names are case sensitive.
If you enter only the WebGate's IP address you will get the home page preloaded by the factory or the DEFAULT.HTML page if it is present.

WARNING. Your home Web page must be named DEFAULT.HTML. If you don't have any Web page named DEFAULT.HTML, the INDEX.HTML factory preloaded page will be served when you will enter the WebGate's IP address.

All parameters of all Carel controllers on the RS485 network can be monitored and modified through the usage of web pages

7.3.1 <%var(x, y, z)%> Visualization Tag

This is a tag that provides access to the network variables' data of the devices connected to the WebGate for monitoring through a Web browser.
To see a particular network variable of a device connected to the WebGate, you must insert the following tag into the HyperText of the HTML page:

<%var(x, y, z)%>

where:
x = device address;
y = variable type;
z = supervision index of the variable.

[**x**]: it is the serial address of the device connected on the local RS485 Carel network. It can change in the range [**1 ÷ maxdevs**], where "maxdevs" is the system parameter, setting the maximum number of devices you want to connect to the WebGate. You can configure the "maxdevs" parameter using a console or a Web browser through the HTML configuration page "RS485".

[**y**]: expresses the variable type to be displayed: digital, analog or integer, as shown in the following table:

y	Type of variable
1	Digital
2	Analogue
3	Integer

Tab.7.3.1

[**z**]: it represents the supervision index of the variable in a communication between the device and a supervisor. This index must be searched for in the CAREL database, which can be supplied by CAREL, of the device connected to the network RS485.

Example 1
To create a simple Web page which will monitor the temperature set-point of the device 3 (x=3), rapresented for example by an analogue (y=2) variable having index 123 (z=123), you must insert the tag: **<%var(3, 2, 123)%>** in the proper position.

Example 1
<HTML> <head> <title>Display temperature setpoint of device 3</title> </head> Temperature Setpoint = <%var(3,2,123)%><p> </HTML>

Tab. 7.3.2

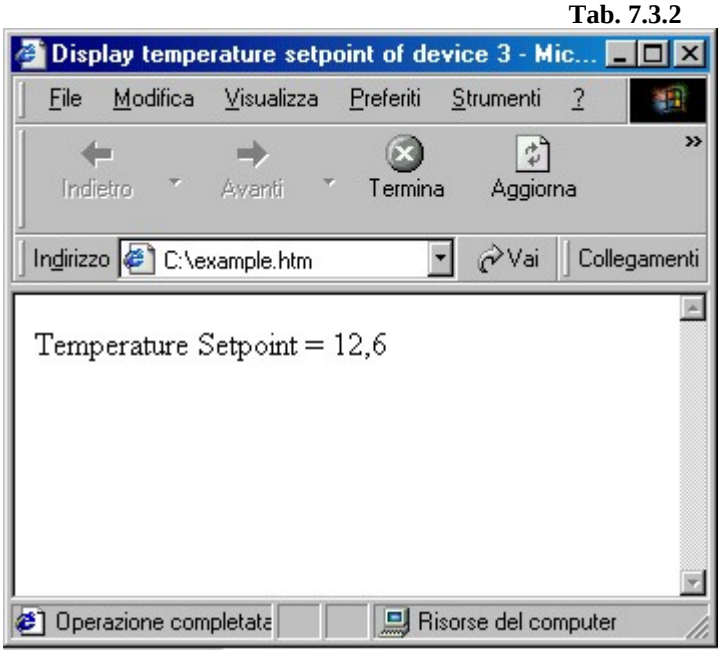


Fig. 7.3.1

Example 2
In Tab.7.3.3 you can see a more complex HTML text for the following page having a figure named “IR32.jpg” and the digital variable: “compressor status” being monitored.

Fig. 7.3.2

Example 2

```
<HTML>

<HEAD>
 <title>compressor status</title>
</HEAD>
•
<BODY bgcolor=”#FFFFFF” text=”#000000”>
<TABLE width=”627” border=”0” cellspacing=”0” align=”center”>
<TR>

<DIV align=”center”>
<FONT face=”Verdana, Arial, Helvetica, sans-serif”>
<img src=”IR32.jpg” width=”257” height=”187” align=”top” vspace=”0” hspace=”0”>
</FONT>
</DIV>

<TABLE width=”346” border=”0” cellspacing=”0” align=”center”>
 <TR>
 <TD width=”255” align=”center” height=”35” bgcolor=”#f8f8f8”>
  <DIV align=”center”>
    <FONT face=”Verdana, Arial, Helvetica, sans-serif”></FONT>
    <FONT face=”Verdana, Arial, Helvetica, sans-serif”>
    <B><font size=”2”></font></B>
•       </FONT>
  </DIV>
  <DIV align=”right”>
    <B><font face=”Verdana, Arial, Helvetica, sans-serif” size=”2”>
       compressor status:
       </font>
    </B>
  </DIV>
 </TD>

 <TD width=”87” height=”35” align=”center” bgcolor=”#e8e8e8”>
 <FONT face=”Verdana, Arial, Helvetica, sans-serif” size=”2” color=”#0000ff”>
   <b><b> <%var(1,1,4)%> </b></b>
 </FONT>
 </TD>
 </TR>
</TABLE>

</TR>
</TABLE>
</BODY>
</HTML>
```

Tab. 7.3.3

WARNING. If the device is in the OFF-LINE status (or the WebGate is busy in reading all the variables of device for the first time), the tag conversion will give back the following warning string: “**Unit OFF-LINE: unreliable value**”

7.3.2 Writing device parameters using Forms

One of the most important aspects of the Ethernet communications between “Web servers” and their “clients” is the possibility to include interactive features. The easiest way to do it is using “forms”.
Forms are used in Web pages to get information from browser end users.
By a form the client can send information to the server by using input objects such as text boxes, menus, check boxes and so on.
In this chapter, after a brief introduction on the basis elements of a form, you will see how forms can be used for reading and writing both WebGate’s parameters and the value of one or more variables of the device connected to the WebGate.

How the contents of a form can be processed by the server once the client submit the information? Generally a program inside the server, as CGI (Common Gateway Interface) or [N/I]SAPI (Netscape/Internet Server Application Program Interface), parses the submitted information data.
As WebGate hasn’t any CGI program on board it parses data in real time using the GET or POST commands, having information data inside the commands themselves.
Once information has been entered into a form, the form is submitted to the server. This is accomplished using the “Submit” function. Alternatively, a form that has not already been submitted can be restored to its original values using the “Reset” function. These two functions, “Submit” and “Reset”, are standard HTML form elements that control forms in Web pages. For information on their use and syntax, please consult a standard HTML reference.

7.3.2.1 The form attributes

The form attributes are **name**, **method** and **action**:

Form Attributes
name
method
action

Tab. 7.3.2.1.1

For example: <form name="form4" method="post" action="mypage.htm">.
The second two attributes, method and action, are the most important to make the form work.

ACTION attribute: Generally the action attribute specify the address of the program that will handle the data inside the form. It is usually set to a URL pointing to a CGI script to decode the data. For example, the code:
<FORM ACTION = "http://www.myserver.com/cgi-prog/post-command…>
is for a script called post-command in the directory cgi-prog on the server which address is given by the URL www.myserver.com.

As WebGate hasn't any CGI program on board to parse the data, the only URLs permitted for this attribute are those pointing to pages previously loaded (included the page with the form itself), able to show the effects of the form data sent by the user.

For example:
ACTION = "IR32cold.HTML" or ACTION = "192.168.10.42/mypage.htm" where 192.168.10.42 is the IP address of the WebGate.

METHOD attribute: This attribute defines the method used to send data to the server.
GET and POST are the two supported methods. ENCTYPE is not supported.
GET is the default method for a client, trough a browser, to submit requests to a server. GET method sends data to the server by appending the data to the URL form itself after a question mark (?) separator.
POST: it is similar to the previous method: the main difference is that data do not come into a unique request, but after the headers of the request itself. Once the WebGate has received a request from a form using POST, it knows it has to continue "listening", after the requested headers, the parameters contained in the form.
for the rest of information data.
The usage of POST instead of GET is reccomended for security reason, because using the GET method all the information appears in the URL textbox of the browser.

NAME attribute: This attribute is important for a CGI program to check data before submission. This isn't our case.

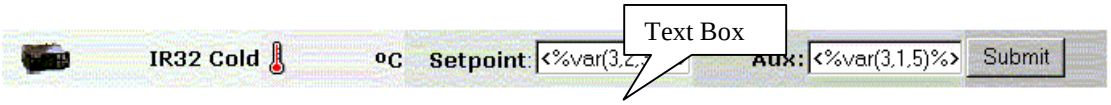
7.3.2.2 The form elements

A form generally has different objects, useful to customise information that will be sent to the server.

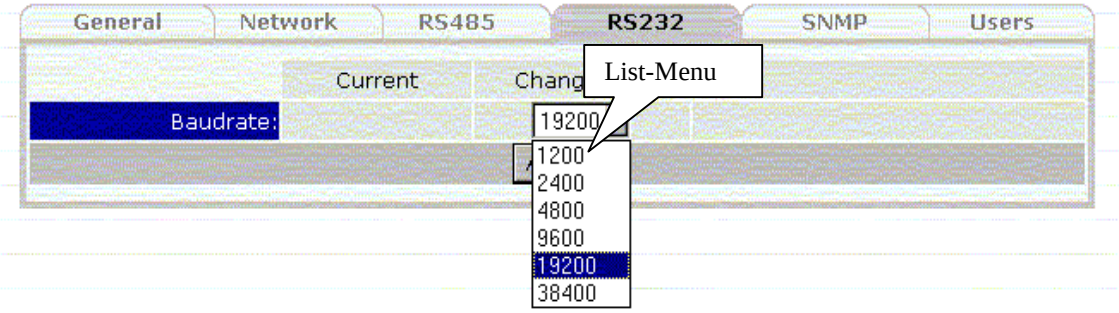
FORM Objects
Text-Box
List-Menu
Jump-Menu
Button
Check-Box
Radio-Button

Tab. 7.3.2.2.1

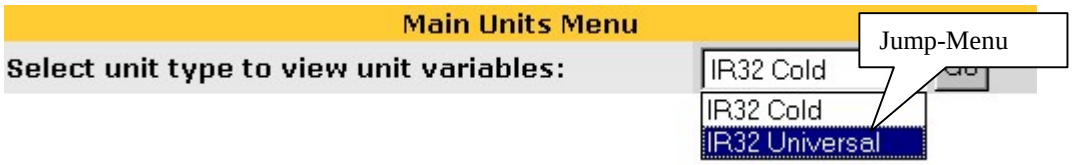
Text-Box: It's a field inside which user can write an alphanumeric text, as passwords, names, numeric values, and so on. For example, use this tag when you want to modify a network variable value.



List-Menu: Useful when you want to present multiple choices in a limited space. An example maybe the selection of the baudrates for serial communication in a set of standard values (1200,2400, 4800, 9600, 19200).



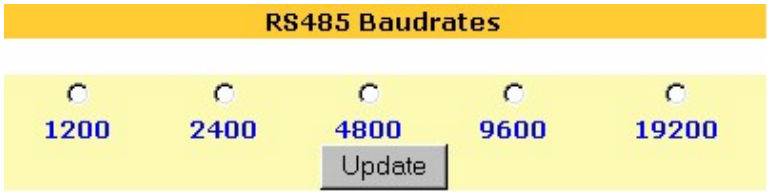
Jump-Menu: A jump menu is a pop up menu that lists options that link to documents or files. This object is interesting when you want a fast menu to select different pages referring to different devices.



Button: Form buttons control form operations. You can use a form button to submit data to a server for processing, or use a form button to reset a form.

Check-Box: Check-boxes allow the user to select more than one option from a set of options. Each check-box object is an individual element and must have a unique name in the Name field.

Radio-Button: The Radio-Button can be seen when you select only one choice from a set of options. Radio buttons are typically used in groups. All radio buttons in a group must have the same name and must contain different field values.



The most used objects in custom application pages are Text-Box, List-Menu, Button and Jump-Menu.
The following paragraphs show some examples of using objects inside a form.

7.3.2.3 Using Text-Boxes and Buttons inside forms

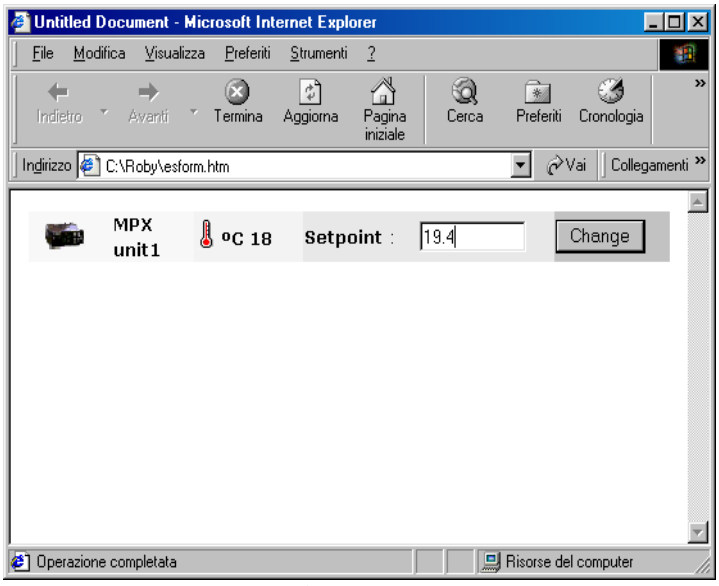


Fig. 7.3.2.3.1

Example of using text-box and button inside a form

```
<HTML>
<head>
<title>Untitled Document</title>
<meta http-equiv="Content-Type" content="text/HTML; charset=iso-8859-1">
</head>
<body bgcolor="#FFFFFF" text="#000000">
<form name="form1" method="post" action="esform.htm">
<table width="500" border="0" cellspacing="0" align="center">
<tr>
 <td width="51" align="center" height="35" bgcolor="#f8f8f8">
 <div align="center">
 <font face="Verdana, Arial, Helvetica, sans-serif">
  <a href="detail.htm" target="main">
   <img src="Unit.gif" width="38" height="20" align="middle" border="0"></a>
 </font></div></td>
 <td width="50" align="right" height="35" bgcolor="#f8f8f8">
 <div align="left">
 <font face="Verdana, Arial, Helvetica, sans-serif">
  <b><font size="2">MPX unit1</font></b>
 </font></div></td>
 <td width="14" align="right" height="35" bgcolor="#f3f3f3">
 <font face="Verdana, Arial, Helvetica, sans-serif">
  <b><img src="r.gif" width="12" height="22"></b>
 </font></td>
 <td width="50" align="right" height="35" bgcolor="#f3f3f3" nowrap>
 <div align="left">
 <font face="Verdana, Arial, Helvetica, sans-serif">
  <b><font size="2" color="#0000FF"><%var(1,2,1)%></font>
     <font size="2">&ordm;C</font></b>
 </font></div></td>
 <td width="20" align="right" height="35" bgcolor="#e8e8e8">
 <font face="Verdana, Arial, Helvetica, sans-serif">
 <b><font size="2">Setpoint</font></b>
 </font>:</td>
 <td width="20" height="35" align="left" bgcolor="#e8e8e8">
 <font face="Verdana, Arial, Helvetica, sans-serif">
 <b><font face="Verdana, Arial, Helvetica, sans-serif">
 <b><font size="2" color="#0000CC">
  <input type="text" name="?script:var(1,2,7,12.4,13.2)" size="10" maxlength="16"    value="<%var(1,2,7)%>">
  </font><font size="2"> </font>
 </b></font></b></font></td>
 <td width="50" height="35" valign="middle" bgcolor="#c2c2c2" align="left">
 <div align="center">
  <input type="submit" name="Button1" value="Change">
 </div></td></tr></table></form></body></HTML>
```

Tab.7.3.2.3.1

Table 7.3.2.3.1 shows the HTML text of a page having a form.
First you can observe the three main attributes of the form into its tag:
<form name="form1" method="post" action="esform.htm">

Note that:

- the form content is a table; the first table data is a figure called “Unit.gif” linked by <a href…> tag to the page detail.htm. Clicking your mouse on the figure from the browser, the WebGate will send the new page detail.htm;
- a following table entry displays a temperature using the tag <%var(1,2,1)%>.
- the <TD> tag before the last includes a Text-Box field used to change the value of the analog variable 7 of the device number 2; 12.4 and 13.2 are the minum and maximum value for that variable:
  <input type="text" **name="?script:var(1,2,7,12.4,13.2)"** ...**value="<%var(1,2,7)%>">**

  This section describes the name and value of this tag.

  name: The name string is composed by two fields: an “header” and a “command field” as you can see in the following table:

Writing tag for text-boxes	
Header	**Command-field**
?script:	generic command or var(x,y,z,min,max)

Tab. 7.3.2.3..2

The header **“?script:**“ is used to signal to WebGate that the user is writing the parameter, when it parses the submitted data.
In this example the parameter is a type[y] variable of device[x] having supervision index[z]; min and max are its minimum and maximum values that define the variation range allowed for that parameter. You can also use one of the other available commands. Refer to the configuration documentation to have a list of the available commands. When the user digits a value for this variable into the text-box, the WebGate parses the command using that value, as if the command has been sent using the RS232 console interface.

value: Setting the value using the “visualization tag”, you can also see the result of the command sent.

The last table data tag <TD> includes the button using:

<input type="**submit**" name="Button1" value="Change">

As the input type is set to “submit” you will submit the data form by clicking on the button.
Once the value has been digitised into the text-box, you can also send it using the enter key (↵).

WARNING: sometimes you may observe a delay during the download of a page specified in the ACTION attribute, after a new value is entered for devices variables. This delay is due to the communication mechanism: WebGate sends to the device, through the RS485 serial line, an appropriate message reporting the new value, and it will wait at most 30 seconds to check if the setting of the new value has occured. If this communication fails, or if the max. time has been exceeded, WebGate will return the HTML page with the old variable value. For example, Fig. 7.3.2.3.2 shows what the browser may display during the communication delay, due to the change of an “IR32” cold setpoint.
Fig. 7.3.2.3.3 report the page after the page download ended.

Fig. 7.3.2.3.2

Fig. 7.3.2.3.3

7.3.2.4 Using List-Menu inside forms

The following table shows the HTML code of a page having a form with a List-Menu inside.

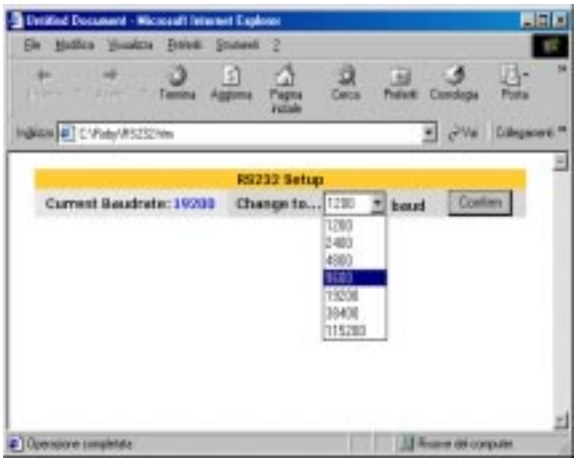


Fig. 7.3.2.4.1

Example of using list-menu inside a form

```
<HTML>
<head>
<title>Untitled Document</title>
<meta http-equiv="Content-Type" content="text/HTML; charset=iso-8859-1">
</head>

<body bgcolor="#FFFFFF" text="#000000">
<form name="form1" method="get" action="RS232.htm">
<table width="517" border="0" cellspacing="0" align="center">
•     <tr align="center">
   <td colspan="5" bgcolor="#FFCC33">
   <font face="Verdana, Arial, Helvetica, sans-serif">
   <b><font size="2">RS232 Setup</font></b></font></td>
  </tr>
  <tr>
   <td width="142" align="right" height="12" bgcolor="#e8e8e8">
   <font face="Verdana, Arial, Helvetica, sans-serif">
   <b><font size="2">Current Baudrate:</font></b></font></td>
   <td width="58" height="12" align="left" bgcolor="#e8e8e8">
   <font face="Verdana, Arial, Helvetica, sans-serif">
   <b><font face="Verdana, Arial, Helvetica, sans-serif">
   <b><font size="2" color="#0000CC"><%Baudrate232%></font>
   <font size="2"> </font></b></font></b></font></td>
   <td width="96" height="12" align="right" bgcolor="#d7d7d7">
   <font face="Verdana, Arial, Helvetica, sans-serif">
   <b><font size="2">Change to...</font></b></font></td>
•     <td width="116" height="12" valign="middle" bgcolor="#d7d7d7" align="left">
•     <font face="Verdana, Arial, Helvetica, sans-serif">
•     <b><font size="2">
    <select name="select2" size="1">
      <option value="?script:baudrate232=1200" selected>1200</option>
      <option value="?script:baudrate232=2400">2400</option>
      <option value="?script:baudrate232=4800">4800</option>
      <option value="?script:baudrate232=9600">9600</option>
      <option value="?script:baudrate232=19200">19200</option>
      <option value="?script:baudrate232=38400">38400</option>
      <option value="?script:baudrate232=115200">115200</option>
    </select>
  baud</font></b></font></td>
   <td width="95" height="12" valign="top" bgcolor="#d7d7d7" align="center">
   <input type="submit" name="Button" value="Confirm">
   </td>
  </tr>
  <tr>
   <td colspan="5" align="center" height="34"></td>
  </tr>
</table></form></body></HTML>
```

Tab. 7.3.2.4.1

As you can see, the second table row <TR> has a <TD> tag to display the current RS232 baudrate by the “Visualization tag” <%Baudrate232%>.
The successive boldface text highlights part of the List-Menu HTML language.
The size element sets the number of items showing at once.
The <**option**> elements specify the actual choices on the menu.
An occurrence of the attribute “selected” in the <option> element sets the form control to select this item by default. If no value is selected, typically the field remains undefined.
Note that the value string has the same structure of the name string for Text Boxes and it is composed by two fields: an “header field” (?script:) and a “command field”.
The header field “?script:“ is used to signal to WebGate that the user is writing the parameter, when it parses the submitted data.

7.3.2.5 Using Jump-Menu inside forms

A jump menu is a pop-up menu listing possible connections that link to documents or files.
In the following table you can see the HTML text of a form using a Jump-Menu.

Fig. 7.3.2.5.1

Example of using jump-menu inside a form

```
<HTML>
<head>
<title>Untitled Document</title>
<meta http-equiv="Content-Type" content="text/HTML; charset=iso-8859-1">
<script language="JavaScript">
<!--
function MM_jumpMenu(targ,selObj,restore){ //v3.0
  eval(targ+".location='"+selObj.options[selObj.selectedIndex].value+"'");
  if (restore) selObj.selectedIndex=0;
}

function MM_findObj(n, d) { //v4.0
  var p,i,x;  if(!d) d=document; if((p=n.indexOf("?"))>0&&parent.frames.length) {
    d=parent.frames[n.substring(p+1)].document; n=n.substring(0,p);}
  if(!(x=d[n])&&d.all) x=d.all[n]; for (i=0;!x&&i<d.forms.length;i++) x=d.forms[i][n];
  for(i=0;!x&&d.layers&&i<d.layers.length;i++) x=MM_findObj(n,d.layers[i].document);
  if(!x && document.getElementById) x=document.getElementById(n); return x;
}

function MM_jumpMenuGo(selName,targ,restore){ //v3.0
  var selObj = MM_findObj(selName); if (selObj) MM_jumpMenu(targ,selObj,restore);
}
//-->
</script>
</head>

<body bgcolor="#FFFFFF" text="#000000">
<form name="form1" method="get" action="E.htm">
<table width="517" border="0" cellspacing="0" align="center">
 <tr align="center">
  <td colspan="5" bgcolor="#FFCC33"><font face="Verdana, Arial, Helvetica, sans-serif">
  <b><font size="2">Main Units Menu </font></b></font></td>
 </tr>
 <tr>
  <td width="300" align="left" height="12" bgcolor="#e8e8e8">
    <font face="Verdana, Arial, Helvetica, sans-serif"><b><font size="2">Select
    unit type to view unit variables:</font></b></font></td>
  <td width="150" height="12" valign="middle" bgcolor="#d7d7d7" align="left">
    <div align="center"><font face="Verdana, Arial, Helvetica, sans-serif">
    <b><font size="2">
```

```
    <select name="menu1" onChange="MM_jumpMenu('parent',this,0)">
      <option value="IRcold.htm" selected>IR32 Cold</option>
      <option value="IRuniv.htm">IR32 Universal</option>
    </select>
  <input type="button" name="Button1" value="Go" onClick="MM_jumpMenuGo('menu1','parent',0)">
  </font></b></font></div></td>
 </tr>
</table></form></body></HTML>
```

Tab. 7.3.2.5.1

The black boldface text highlights the <SCRIPT> option, used to edit the three JavaScript functions for the Jump-Menu. It has been automatically created by the HTML editor used to construct this page.
The following text is relative to the pop-up menu.
Figures 7.3.2.5.1 and 7.3.2.5.2 show the effects in your browser, if it is able to make JavaScript running.

Fig. 7.3.2.5.2

IR32 Cold is the default item, as you can see in the first <**option**> elements of the highlighted text.
When you click on the Go button, a link to the IR32cold.htm page is activate, and the WebGate sends that page to your browser.
In the same way, by selecting the second item on Jump-Menu, the IR32univ.htm page appears.

WARNING. The use of Java or JavaScript sometimes can't give the expected results. They could not work properly depending mainly by the version of the browser you are using. Generally, they works fine on Microsoft® Internet Explorer™ version 5 or higher.

7.3.2.6 Using Check-Boxes inside forms

A check-box allows you to select more than a option from a set.
The following table gives you an example about the choice of a serial baudrate.

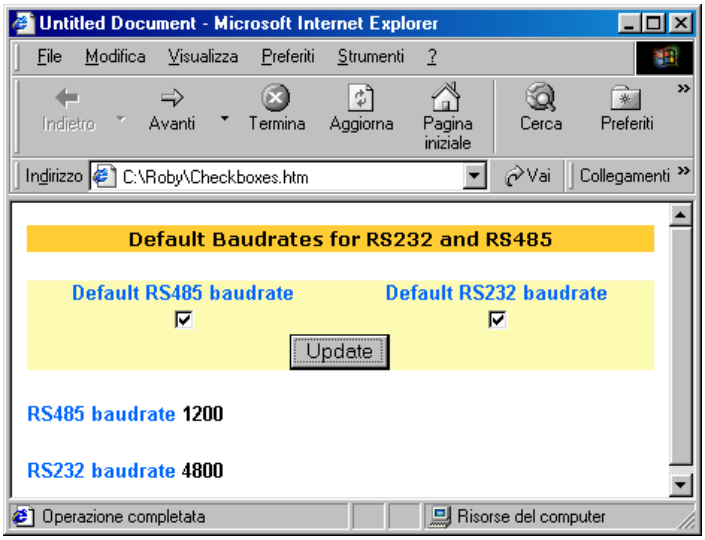


Fig. 7.3.2.6.1

Example of using check-boxes inside a form

```
<HTML>
<head>
<title>Untitled Document</title>
<meta http-equiv="Content-Type" content="text/HTML; charset=iso-8859-1">
</head>
<body bgcolor="#FFFFFF" text="#000000">
<table width="100%" border="0" cellspacing="0" align="center">
  <tr align="center"><td colspan="5" bgcolor="#FFCC33"><font face="Verdana, Arial, Helvetica, sans-serif">
    <b><font size="2">Default Baudrates for RS232 and RS485</font></b></font></td>
  </tr>
</table>
<form name="form1" method="post" action="Checkboxes.htm">
 <table width="100%" border="0" cellspacing="0" cellpadding="0" bordercolor="#FDFBB3" bgcolor="#FDFBB3">
  <tr><td>
    <div align="center"><b><font face="Geneva, Arial, Helvetica, san-serif" size="2" color="#0066FF">
      Default RS485 baudrate</font></b></div>
   </td><td>
    <div align="center"><b><font face="Geneva, Arial, Helvetica, san-serif" size="2" color="#0066FF">
      Default RS232 baudrate</font></b></div>
   </td></tr><tr><td>
    <div align="center"><input type="checkbox" name="RS485" value="?script:Baudrate485=19200">
   </div>
   </td><td>
    <div align="center"><input type="checkbox" name="RS232" value="?script:Baudrate232=19200">
   </div>
   </td></tr>
 </table>
 <table width="100%" border="0" cellspacing="0" cellpadding="0" bgcolor="#FDFBB3">
  <tr>
   <td>
    <div align="center"><input type="submit" name="Submit" value="Update"></div>
   </td>
  </tr>
 </table>
</form>
<p><b><font face="Geneva, Arial, Helvetica, san-serif" size="2" color="#0066FF">RS485
 baudrate</font></b>
 <%baudrate485%>
</p>
<p><b><font face="Geneva, Arial, Helvetica, san-serif" size="2" color="#0066FF">RS232
 baudrate</font></b>
 <%baudrate232%>
</p>
</body>
</HTML>
```

Tab. 3.2.6.1

Each checkbox you add to a form must have a unique name.
About its value, the rule is the same enounced for text-boxes.
Because the check-box option is normally used for choosing a value in a range, the command generally is a writing command. In this example it is **“Baudrate232=19200”** and **“Baudrate485=19200”**.
Of course you can choose one or both of the above mentioned commands, than confirm clicking on the Update button or typing enter (↵).
The two visualization tag at the end allow you to see the settings.
The above figure shows the case of both options selected, when the first baudrate was 1200 and the second 4800.
The following figure displays the result, if ACTION is set to the same page.

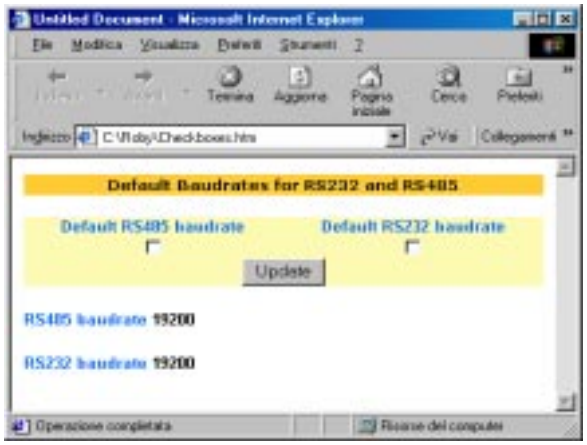


Fig. 7.3.2.6.2

7.3.2.7 Using Radio-Buttons inside a form

The Radio-Buttons are useful when the user must select only one choice from a set of options.
The Radio-Buttons are typically used in groups. All the Radio-Buttons in a group must have the same name and must contain different field values.
The boldface text for a group of five buttons in the next table allows the user to select a unique baudrate in a range of five possibilities.
The Radio-Buttons elements are inserted in a table 1x5 (note the <TD> tag).

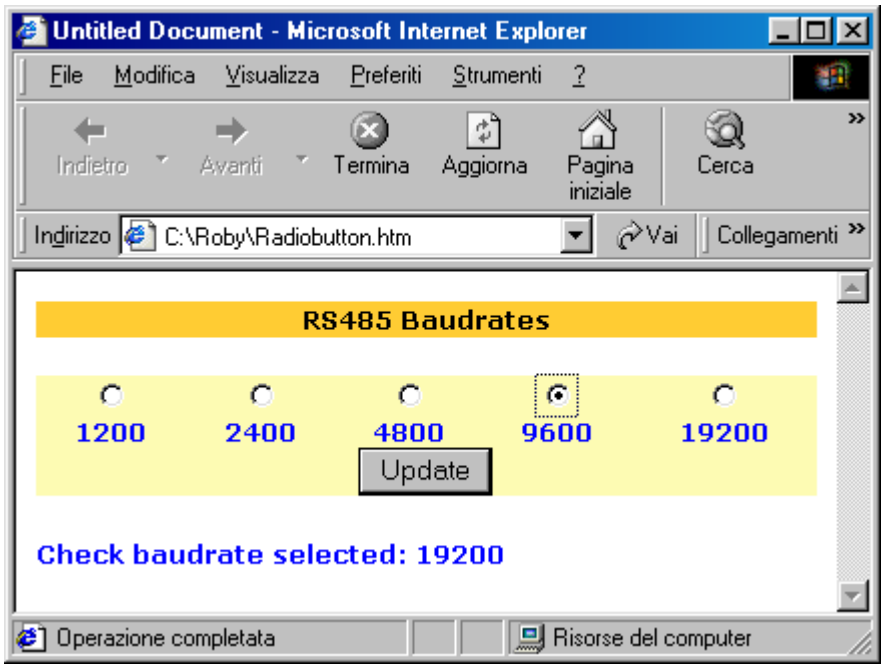


Fig. 7.3.2.7.1

Group of five Radio-Buttons

……
……
  <td colspan="5">
 	
  <input type="radio" name="RS485" value="?script:Baudrate485=1200">
 	
  </td>
  <td>
 	
  <input type="radio" name="RS485" value="?script:Baudrate485=2400">
 	
  </td>
  <td>
 	
  <input type="radio" name="RS485" value="?script:Baudrate485=4800">
 	
  </td>
  <td>
 	
  <input type="radio" name="RS485" value="?script:Baudrate485=9600">
 	
  </td>
  <td>
 	
  <input type="radio" name="RS485" value="?script:Baudrate485=19200">
 	
  </td>
……
……

Tab. 7.3.2.7.1

The name **"RS485"** is the same for all.
The above figure shows the page displayed when the user selects 4800 baud and the current baudrate is 19200. The next figure displays the result.
Note how all the data submitted by the form appear into the URL box of the browser. The ACTION element is GET.

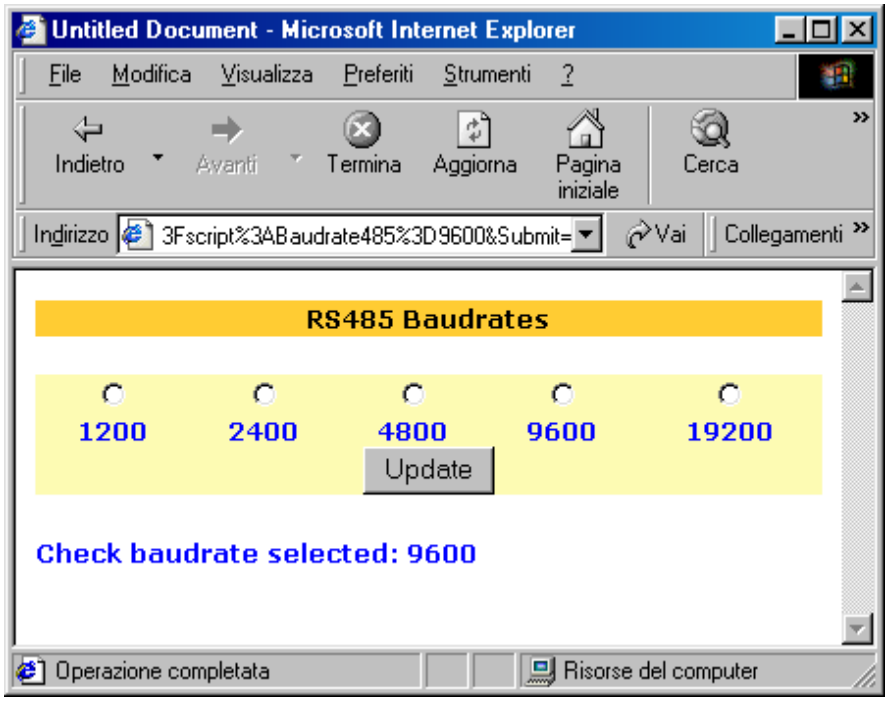


Fig. 3.2.7.2

7.3.3 Changing the IP address and the SUBNET-MASK

All devices on a network requires a unique IP address which host can use to communicate with them. The SUBNET mask, along with the IP address, defines what range of IP addresses are on the local Ethernet LAN.
The user has 2 ways to change the WebGate's IP address and SUBNET mask: the first is from the console using HyperTerminal software, the second is during an Ethernet connection.
An example of the last case is given in the figure and table below.

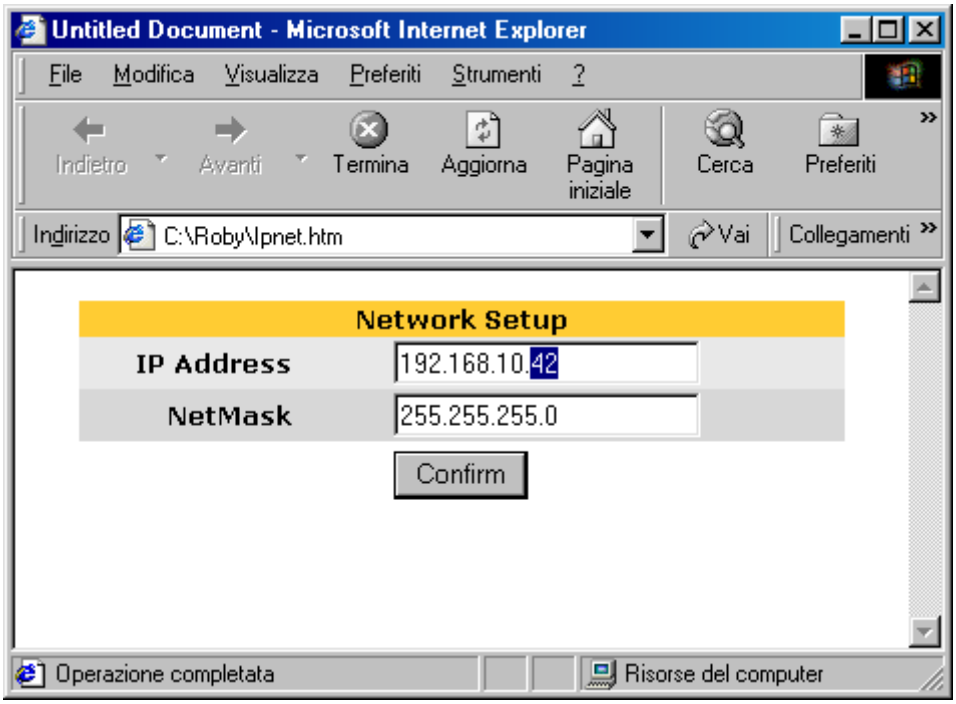


Fig. 7.3.3.1

Example of HTML code for setting IP and NETMASK

```
<html>
<head>
<title>Untitled Document</title>
<meta http-equiv="Content-Type" content="text/html; charset=iso-8859-1">
</head>

<body bgcolor="#FFFFFF" text="#000000">
<form name="form1" method="post" action="Ipnet.htm">
  <table width="382" border="0" cellspacing="0" align="center">
    <tr align="center">
      <td colspan="2" bgcolor="#FFCC33"><font face="Verdana, Arial, Helvetica, sans-serif">
            <b><font size="2">Network Setup </font></b></font></td>
    </tr>
    <tr>
      <td width="105" align="right" height="12" bgcolor="#E8E8E8">
            <font face="Verdana, Arial, Helvetica, sans-serif"><b><font size="2">
            IP Address </font></b></font></td>
      <td width="130" height="12" valign="top" bgcolor="#E8E8E8">
            <font face="Verdana, Arial, Helvetica, sans-serif"><b><font size="2">
        </font><font face="Verdana, Arial, Helvetica, sans-serif"><b><font size="2">
        <input type="text" name="?script:IPaddress" maxlength="16" value="<%IPAddress%>">
        </font></b></font></b></font></td>
    </tr>
    <tr>
      <td width="105" align="right" bgcolor="#D7D7D7">
            <font face="Verdana, Arial, Helvetica, sans-serif">
            <b><font size="2">NetMask</font></b></font></td>
      <td width="130" bgcolor="#D7D7D7">
            <font face="Verdana, Arial, Helvetica, sans-serif"><b><font size="2">
        </font><font face="Verdana, Arial, Helvetica, sans-serif">
            <b><font face="Verdana, Arial, Helvetica, sans-serif"><b><font size="2">
        <input type="text" name="?script:NetMask" maxlength="16" value="<%NetMask%>">
        </font></b></font></b></font></b></font></td>
    </tr>
    <tr>
      <td colspan="2" align="center" height="34">
        <input type="submit" name="Button2" value="Confirm">
      </td>
    </tr>
  </table>
</form>
</body>
</html>
```

Tab. 7.3.3.1

As you can see looking at the boldface text, the rules to insert a text box interface for setting the two mentioned parameters are always the same.
When you change its subnet-mask and/or IP, the WebGate displays the values just selected; however, if some connection process is active (FTP or HTTP) because of other users, a delay is possible before the settings have effect due to the fact that all the connection processes referred to the precedent IP value must be solved by the WebGate.
That’s why, activating a connection using the new IP value during this time interval, you could get the “URL not found” warning or a delay before the request is accepted.
To change the IP faster, it’s better to perform this action when other user’s requests are in stand-by.

7.3.4 Warnings pages

These pages are sent by the WebGate to give a “warning message” when the user tries to do a temporary inhibited or absolutely not permitted operation.

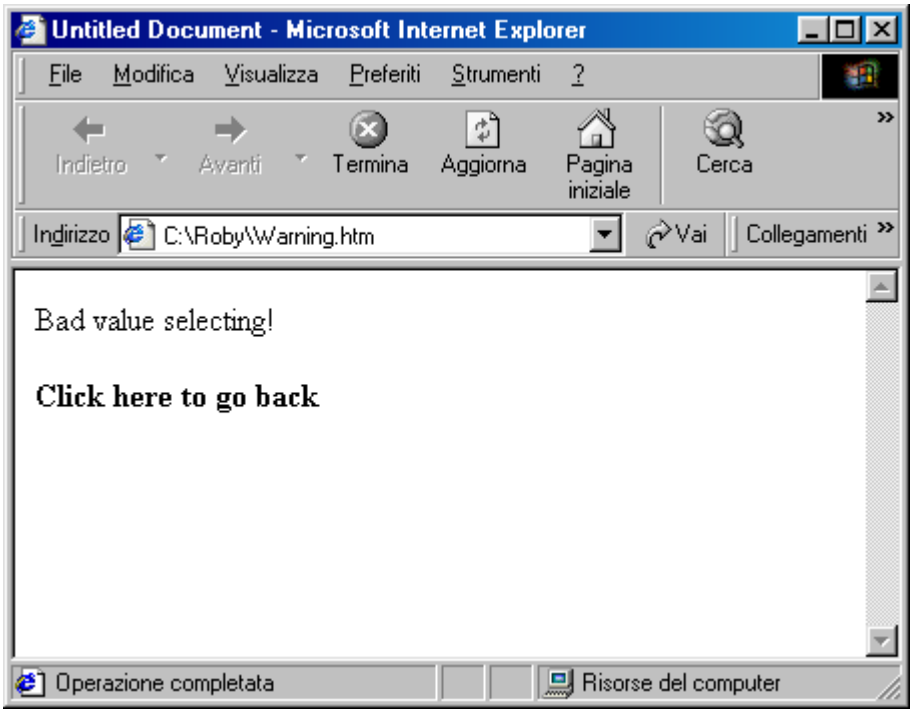


Fig. 7.3.4.1

If your browser is javascript-enabled clicking on the boldface “**click here to go back**” you will return to the previous page.

7.4 Suggestions for HTML pages optimization

HTML pages are text files and, since of this, they uses an appreciable quantity of file space. A big page reduces the amount of files that can be stored into WebGate and ultimately slows down the download.

To reduce the memory wasted, remove any indentation from the file before upload to WebGate. Generally, HTML editors can be configured to automatically obtain the minimum file footprint.

Additionally, use cascading style sheet (CSS) files when more elements in a page or more pages shares the same visualization styles.

Furthermore, use compressed images if possible. For instance, remember that GIF, TIFF and BMP files occupies considerably more space than JPEG format.

8. WebGate SNMP Protocol

8.1 A brief overview of the SNMP protocol

SNMP (**S**imple **N**etwork **M**anagement **P**rotocol) is an Internet-standard protocol, introduced in 1988, for managing devices on IP networks. Network complexity has increased very hard in last years by addition of several different kinds of devices, as routers, servers, proxies, switches, workstations, printers, gateways, UPSs, and so on.
SNMP can be used to monitor both standard network hardware and specific variables, like the temperature and humidity inside a room.
The core of the protocol is a simple set of operations that gives to the network administrators the ability to control and change the state of some SNMP-based devices or the value of the most important variables of a controller, i.e. a setpoint.
The two principal entities of an SNMP network architecture are managers and agents (Fig.8.1.1). A **manager** is a server system that can handle management tasks for a network. Managers are often referred to as *Network Management Station* (NMSs). An NMS is responsible for polling agents and for receiving traps from agents in the network. Polling action by NMS is the act of querying an agent for information. A trap is a way for the agent to inform NMS about a particular event.
Generally querying is synchronous, that is agents respond only after a request by NMS. Trap messages are asynchronous: agents send a trap without any request by NMS, because NMS can't knows nothing about the nature of the event and instant the event occur. In other words, there are no restrictions on when the NMS can query the agent or when the agent can send a trap.

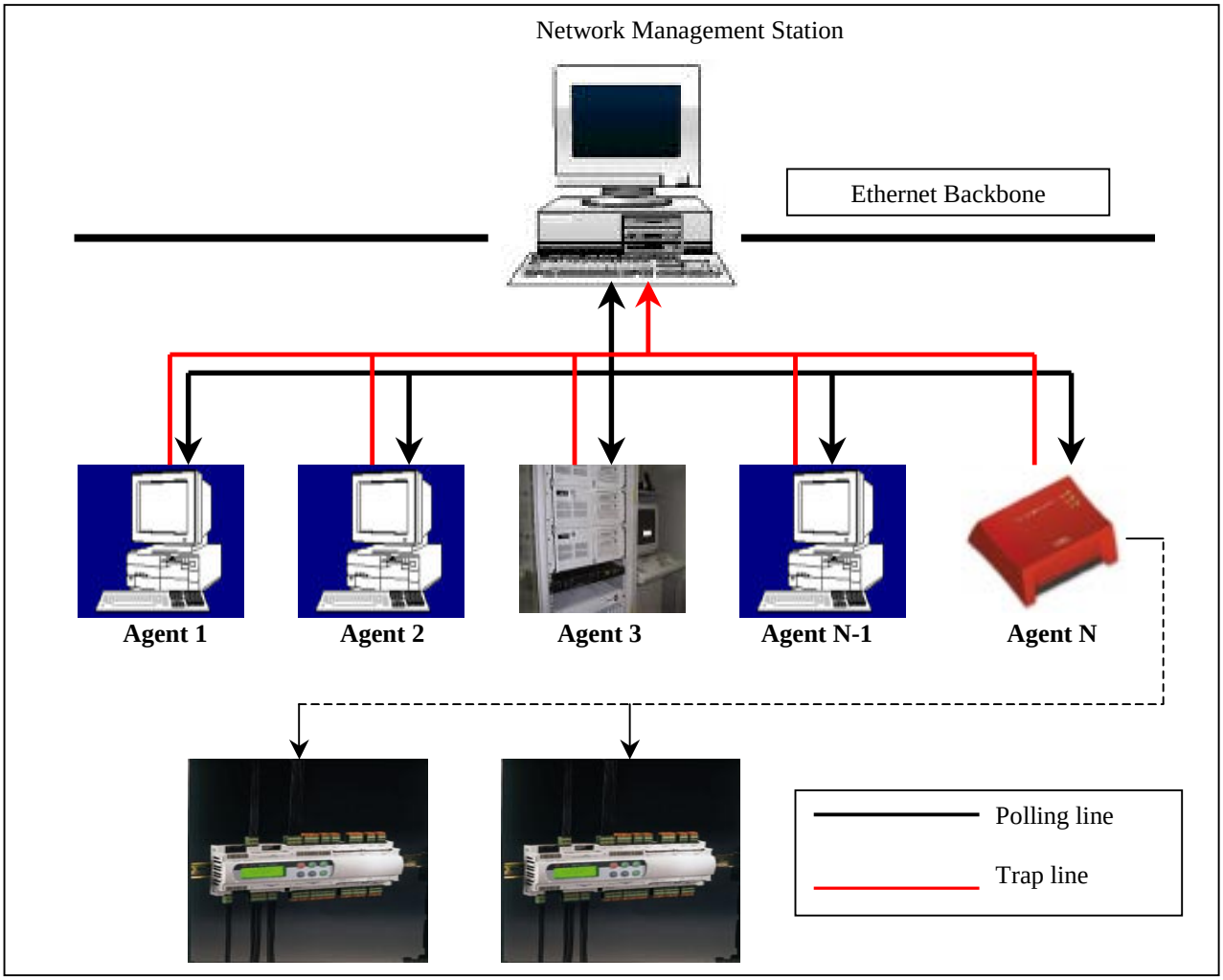


Fig. 8.1.1

Agent is a particular software that runs on the network device (WebGate in our case). It provides management information to the NMS by keeping track of various aspects of the device, such as its state, value of important variables, statistical informations, and so on.

8.2 The structure of management information: agent MIBs

The database, controlled by the SNMP agent, is referred to as the SNMP Management Information Base (MIB), i.e. database of the management information, and is a standard set of objects representing statistical and control values. SNMP additionally allows the extension of these standard values with values specific to a particular agent through the use of private MIBs. Each MIB object is composed by two main attributes (Fig.8.2.1):

MIB object

Name (OID)	**Value**

Fig. 8.2.1

The **Name or Object IDentifier (OID)** uniquely defines a managed object on the network.
Object identifier commonly appear in two forms: numeric or “readable”, as we’ll see in the next paragraphs. It can be thought of as an address that specifies uniquely where is the object on the network.
Value represents the value of the variable associated to the object.

An agent may implement many MIBs. The global set of objects of an SNMP agent is named **MIBview**.

8.3 Naming OIDs: the tree hierarchy structure of the web

Managed objects are organized into a tree-like hierarchy. This structure is the basis for SNMP’s naming scheme.
An OID is made up of a series of integers based on the nodes in the tree, separated by dots (.).
Moreover there’s a human-readable form that’s more friendly than a string of number. This form is a series of names separated by dots, each of which represents a node of the tree. So we can use the number themselves or a sequence of names that represent the numbers.
The main nodes of the tree are the following (Fig.8.3.1).

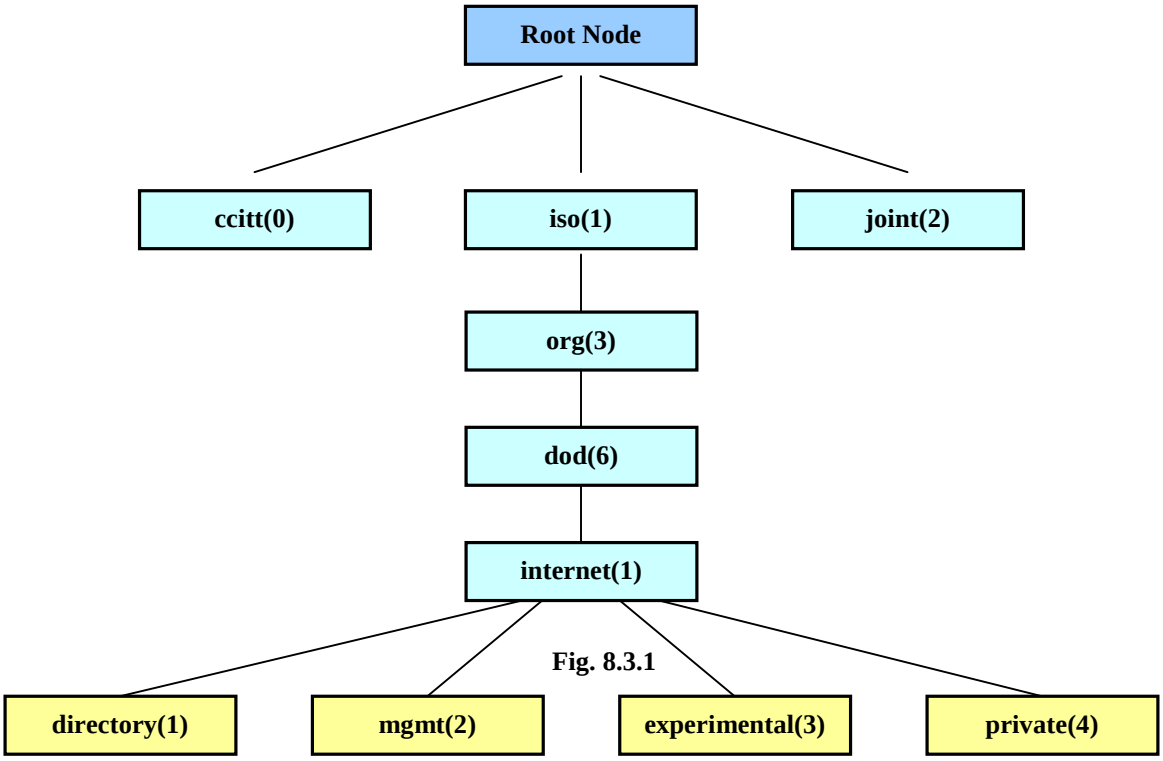


Fig. 8.3.1

The top of the tree is called *root node*. Childrens of a node are called *subtrees*. For example, as figure 8.3.1 shows, the root has three subtrees, that are “*ccitt(0)*”, “*iso(1)*”, “*joint(2)*”. Iso(1) is the only node that contains a subtree. The other two nodes are often called “leaf nodes”. A brief description about the meaning of this nodes can be found in Tab. 8.3.1.

Subtrees of Root-Node	
ccitt(0)	Subtree administered by International Telegraph and Telephone Consultative Committee
iso(1)	International Organization for Standardization
joint(2)	Administered jointly by ISO and CCITT

Tab. 8.3.1

Dualism between series of integer and series of names is now immediate: each node is numbered and named; so internet subtree can be referred to as the sequence ***internet=1.3.6.1*** or ***internet=iso(1).org(3).dod(6).1***
We can define the subtree represented into Fig.8.3.1 using the following syntax:

```
internet      OBJECT IDENTIFIER::= {iso(1).org(3).dod(6).1}={1.3.6.1}
directory     OBJECT IDENTIFIER::= {internet 1}={1.3.6.1.1}
mgmt          OBJECT IDENTIFIER::= {internet 2}={1.3.6.1.2}
experimental  OBJECT IDENTIFIER::= {internet 3}={1.3.6.1.3}
private       OBJECT IDENTIFIER::= {internet 4}={1.3.6.1.4}
```

OBJECT IDENTIFIER is one of the main datatypes used in SNMP protocol and **::=** is a definition operator.

Looking at Fig. 8.3.1we can say that internet is a subtree of the Department Of the American Defence: in fact it is a subtree of the node *dod(6)* in the tree structure of the web.

8.4 Carel Enterprise SNMP Tree

Using the concept of tree-hierarchy, now we can show how the information about snmp-agent and unit-variables are organised by CAREL.

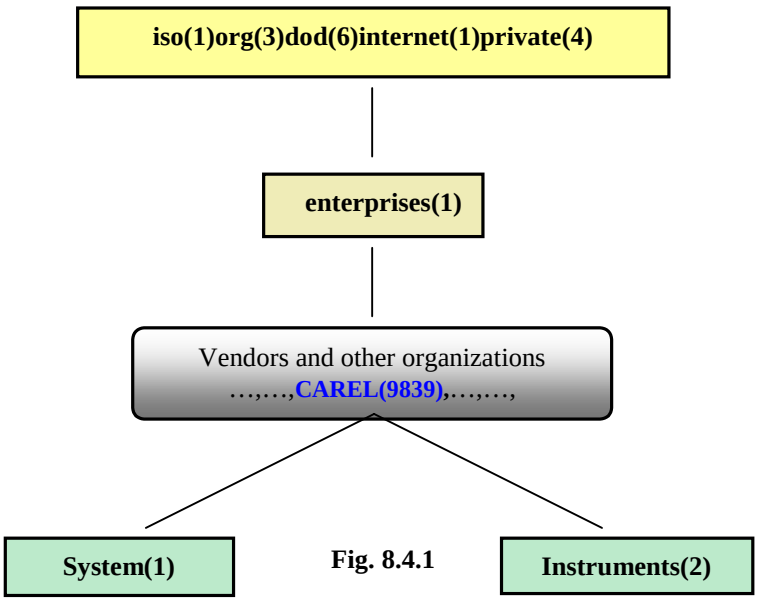


Fig. 8.4.1

Under the node *enterprises::={private(4).1}* there are all the SNMP nodes referring to vendors and private organizations. CAREL is one of them and its OID, assigned by IANA (Internet Assigned Numbers Authority), is ***9839***.

CAREL main subtrees are *System* and *Instruments*: consequently, using the previous syntax:

Carel OBJECT IDENTIFIER::={enterprises 9839}
Carel-System OBJECT IDENTIFIER::={Carel 1}
Carel-Instruments OBJECT IDENTIFIER::={Carel 2}

8.4.1 The Carel-System subtree

Carel-System Subtree has only two objects: **Agent-release** and **Agent-code**.
Agent Release represents the firmware release of the agent firmware incorporated into the WebGate.
Agent Code is the *Agent-identifier* (always **"1"** for WebGate).
The subtree is represented in Fig. 8.4.1.1.

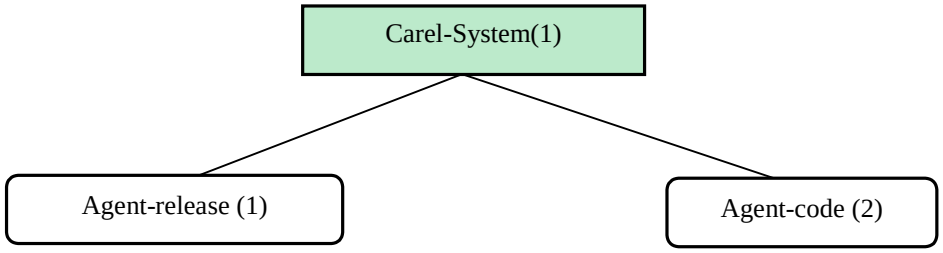


Fig. 8.4.1.1

We can use the following formal template to define these MIB objects:

Agent-release OBJECT-TYPE
SYNTAX INTEGER
ACCESS read-only
Status ---
DESCRIPTION "firmware release of agent"
::={Carel-System 1}

- **Agent-code** **OBJECT-TYPE**
SYNTAX INTEGER
ACCESS read-only
Status ---
DESCRIPTION "agent-identifier"
::={Carel-System 2}

Consequently, if for example a NMS sends a reading command Get request (see **SNMP commands and version**) for the "Agent release", the complete OID is:
{**iso(1).org(3).dod(6).internet(1).private(4).enterprises(1).carel(9839).carel-system(1).1.0**} or
{**1.3.6.1.4.1.9839.1.1.0**}.
Please note that the complete OID ends by 0. The final 0 represents the "instance-identifier" of the OID. Each OID can be though of as the union of two parts: the object-name and the instance-identifier:

OID = Object Name . Instance Identifier

For scalar objects (*not* tables or lists) the instance identifier is always 0.
As the WebGate handles every object as "scalar", every OID ends with ".0".

8.4.2 The Carel-Instruments subtree

Under the node *Carel-Instruments* there are two main subtree: **Carel_WebGate-Info** and **Carel_Units-Var** (Fig. 8.4.2.1).

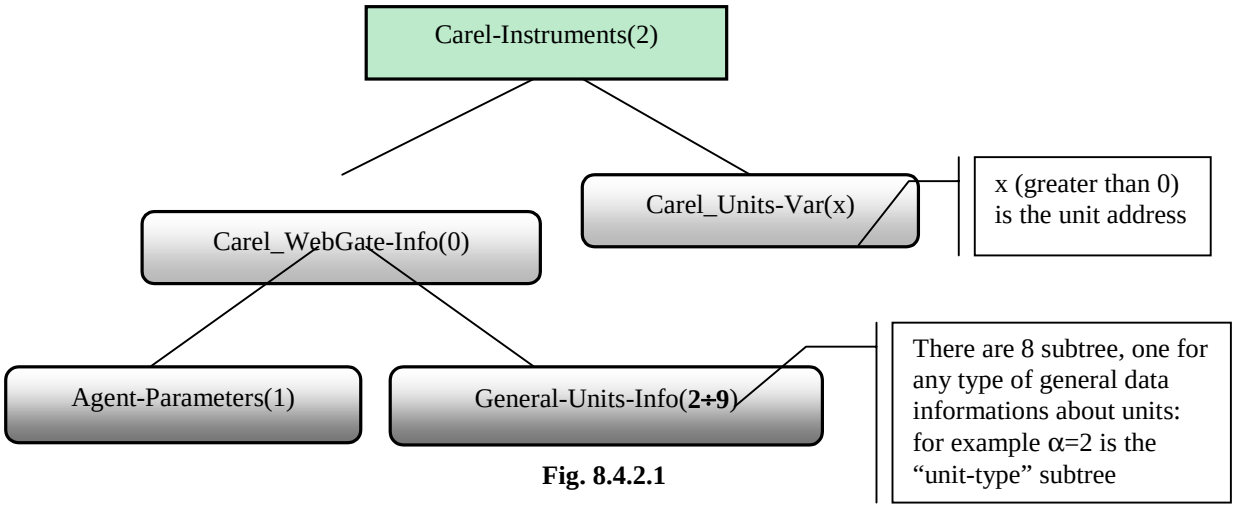


Fig. 8.4.2.1

In the **Carel_WebGate-Info** sub-tree there are information regarding the WebGate's parameters and general data about units (type, code, software release,…)
In the **Carel_Units-Var** sub-tree there are information regarding units' variables.

So we have:

Carel_WebGate-Info	OBJECT IDENTIFIER::={Carel-Instruments 0}	
Carel_Units-Var	OBJECT IDENTIFIER::={Carel-Instruments **α**}	$1 \le \alpha \le$ MaxDevs
Agent-Parameters	OBJECT IDENTIFIER::={Carel-WebGate-Info 1}	
General-Units-Info	OBJECT IDENTIFIER::={Carel-WebGate-Info **β**}	$2 \le \beta \le 9$

Resuming, moving on carel-instruments subtree (OID-NAME=1.3.6.1.4.1.9839.2), we can have the following cases (Table 8.4.2.1).

OID-NAME=1.3.6.1.4.1.9839.2.α.β		
α	**β**	**subtree**
0	1	Agent-parameters
0	$2 \le \beta \le 9$	General-Units-info
$1 \le \alpha \le$ MaxDevs	---	Units-var

Table 8.4.2.1

8.4.3 The Agent-parameters node

There are two parameters under this node (Figure 8.4.3.1): *Netsize* and *Baudrate*, that have the following definition:

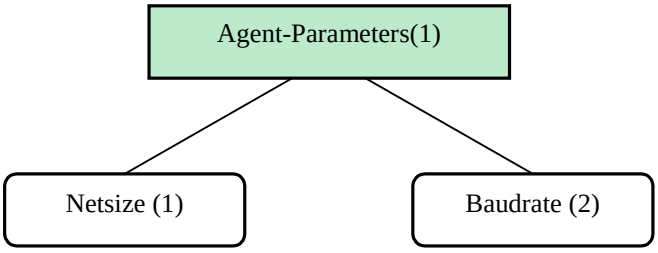


Fig.8.4.3.1

Netsize OBJECT-TYPE
SYNTAX INTEGER
ACCESS read-write
Status ---
DESCRIPTION "max number of units connected to the WebGate"
::={Agent-Parameters 1}

Baudrate OBJECT-TYPE
SYNTAX INTEGER
ACCESS read-write
Status ---
DESCRIPTION "Baudrate of units network"
::={Agent-Parameters 2}

We obtain the complete OID for both the parameters adding to the OID-name the instance 0. So we have:

Netsize OID = 1.3.6.1.4.1.9839.2.0.1.1.0
Baudrate OID = 1.3.6.1.4.1.9839.2.0.1.2.0

8.4.4 The General-Units-info subtrees

Under *Carel_WebGate-Info* node there are other 8 subtree, having general information about units, such as unit-type, unit-code, and so on (Fig. 8.4.4.1). Tab. 8.4.4.1 gives the meanings of each node index.

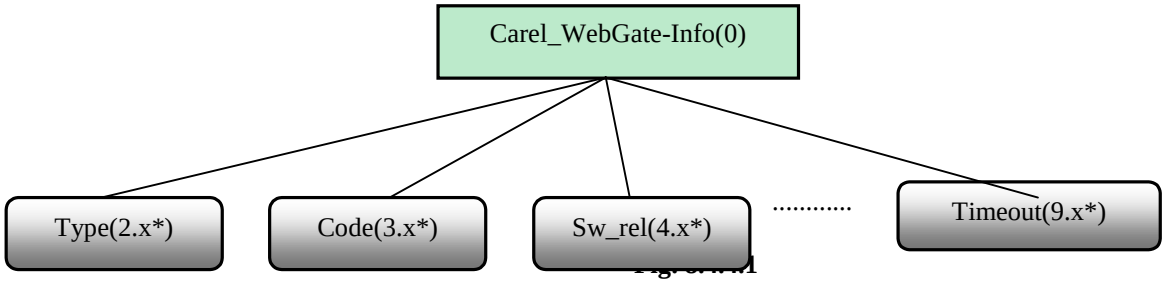


Fig. 8.4.4.1

(*): x (greater than 0) is the unit address.

General-Units-Info subtrees	
Node index	**Data**
2	Unit-Type
3	Unit-Code
4	Unit-Software_release[1]
5	Unit-Min_Software_release[1]
6	Unit-Max_Software_release[1]
7	Unit-No_Answer_counter
8	Unit-Error_checksum_counter
9	Unit-Timeout_counter

Tab. 8.4.4.1

([1]): only for parametric controllers.
Each of the 8 nodes has up to *Netsize* objects (1≤ x ≤Netsize).

Example: Referring to the Unit-Type node (index=2), the situation for 3 units is the following:

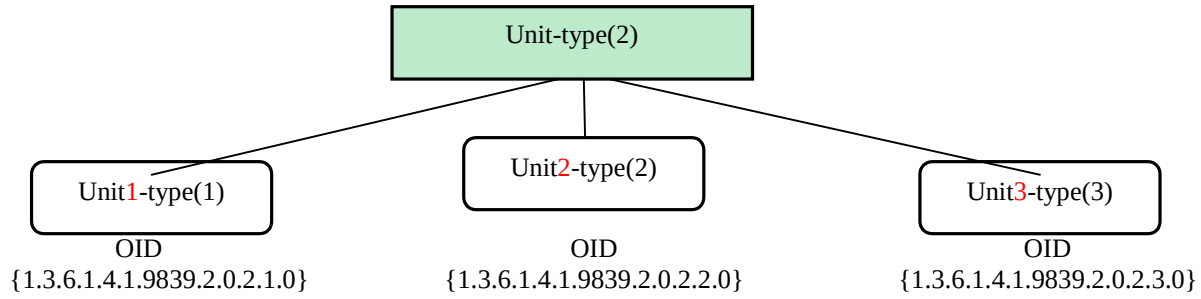


Fig. 8.4.4.2

Here follows the definition for any type of data object under the eight subtrees.

UnitType(X) OBJECT-TYPE
SYNTAX DISPLAY-STRING
ACCESS read-only
STATUS ---
DESCRIPTION "describes the type of the Unit having address X"
::={ Carel-WebGateInfo 2.X}

UnitCode(X) OBJECT-TYPE
SYNTAX INTEGER
ACCESS read-only
STATUS ---
DESCRIPTION "represents the code of the Unit having address X"
::={ Carel-WebGateInfo 3.X}

UnitSoftRel(X) OBJECT-TYPE
SYNTAX INTEGER
ACCESS read-only
STATUS ---
DESCRIPTION "represents the software release of the Unit having address X"
::={ Carel-WebGateInfo 4.X}

UnitMinSoftRel (X) OBJECT-TYPE
SYNTAX INTEGER
ACCESS read-only
STATUS ---
DESCRIPTION "represents the minimum Software release of the Unit having address X"
::={ Carel-WebGateInfo 5.X}

UnitMaxSoftRel (X) OBJECT-TYPE
SYNTAX INTEGER
ACCESS read-only
STATUS ---
DESCRIPTION "represents the maximum Software release of the Unit having address X"
::={ Carel-WebGateInfo 6.X}

UnitNoAnswcnt(X) OBJECT-TYPE
SYNTAX INTEGER
ACCESS read-only
STATUS ---
DESCRIPTION "No answer counter of the Unit having address X"
::={ Carel-WebGateInfo 7.X}

UnitChkErrcnt(X) OBJECT-TYPE
SYNTAX INTEGER
ACCESS read-only
STATUS ---
DESCRIPTION "checksum error counter of the Unit having address X"
::={ Carel-WebGateInfo 8.X}

UnitTimeoutcnt (X) OBJECT-TYPE
SYNTAX INTEGER
ACCESS read-only
STATUS ---
DESCRIPTION "timeout counter of the Unit having address X"
::={ Carel-WebGateInfo 9.X}

8.4.5 The Carel-Units-var subtree

Under this node a manager can find all the SNMP objects relative to the variables of the devices connected to the WebGate. Figure 8.4.5.1 shows how this subtree is structured, assuming Netsize=4.

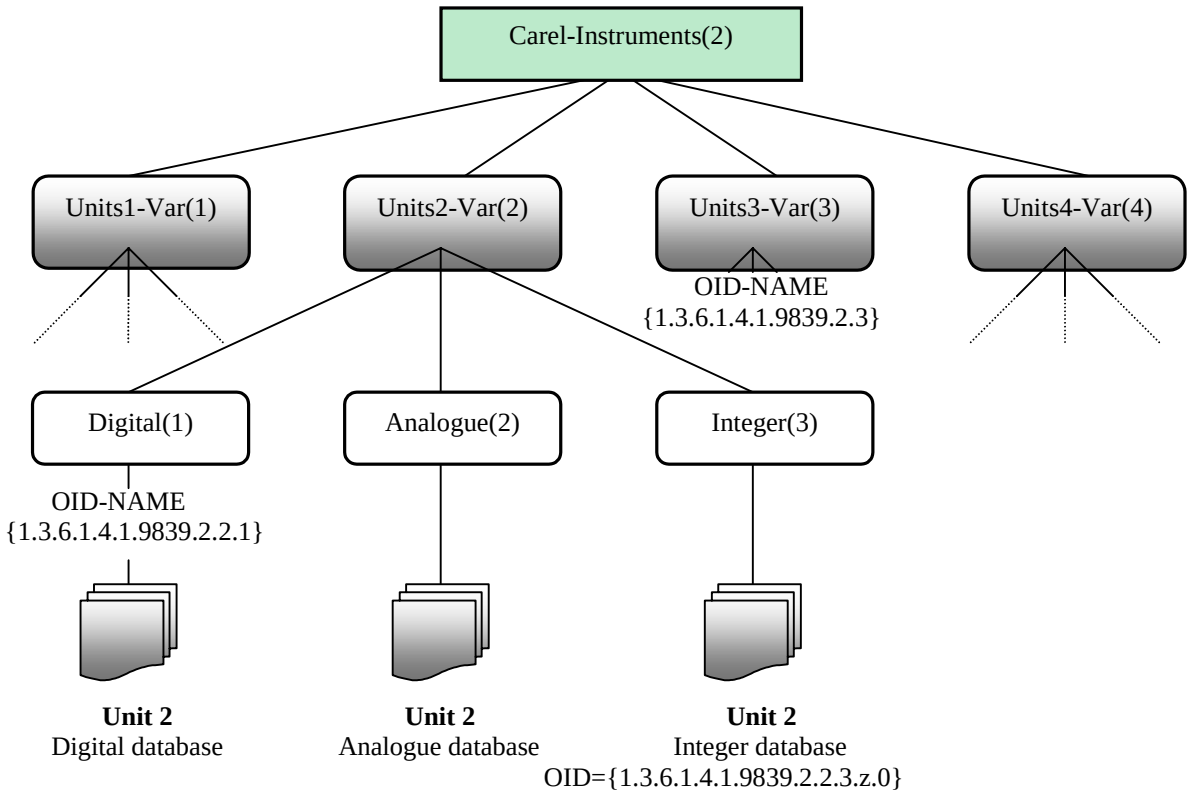


Fig. 8.4.5.1

The formal SNMP definition for an object associated to a variable is:

Variable **OBJECT-TYPE**
SYNTAX INTEGER
ACCESS read-write(1)
Status ---
DESCRIPTION “generic integer, digital or analogue device variable”
::={Carel-Instruments x.y.z}

(1): all variables have default *read-write* access. If a unit parameter is declared “readable only”, when a manager tries a Set-operation, WebGate returns the error message “readONLY”.

Where:

x=unit address	$1 \le x \le$ Netsize
y=type	1=digital, 2=analogue, 3=integer
z=index	$1 \le z \le$ MaxIndex=207

As indicated in Fig. 8.4.5.1, the OID of the first integer variable for unit 2 is 1.3.6.1.4.1.9839.2.2.3.1.0, where unit address=2, type=3 and index=1.

Table 8.4.5.1 gives the relationship between the object OID-name for a generic variable and x,y,z parameters.

OID-name for unit variables*		
OID-name=1.3.6.1.4.1.9839.2.x.y.z		
x=unit address	y=type	z=index
$1 \le x \le$ Netsize	1=digital, 2=analogue, 3=integer	$1 \le z \le 207$

Tab. 8.4.5.1

(*): remember that OID is made by **OID-name.Instance-identifier**.
So the complete OID is always **OID-name.0**

Examples:

Addressing unit n.1

Carel Variable Type	Carel Address	OID-name for variables
Digital	1	1.3.6.1.4.1.9839.2.1.1.1.0
Digital	2	1.3.6.1.4.1.9839.2.1.1.2.0
Digital	....	...
Digital	206	1.3.6.1.4.1.9839.2.1.1.206.0
Digital	207	1.3.6.1.4.1.9839.2.1.1.207.0
Analogue	1	1.3.6.1.4.1.9839.2.1.2.1.0
Analogue	2	1.3.6.1.4.1.9839.2.1.2.2.0
Analogue	...	...
Analogue	206	1.3.6.1.4.1.9839.2.1.2.206.0
Analogue	207	1.3.6.1.4.1.9839.2.1.2.207.0
Integer	1	1.3.6.1.4.1.9839.2.1.3.1.0
Integer	2	1.3.6.1.4.1.9839.2.1.3.2.0
Integer	...	...
Integer	206	1.3.6.1.4.1.9839.2.1.3.206.0
Integer	207	1.3.6.1.4.1.9839.2.1.3.207.0

Tab. 8.4.5.2

Addressing unit n.2

Carel Variable Type	Carel Address	OID-name for variables
Digital	1	1.3.6.1.4.1.9839.2.2.1.1.0
Digital	2	1.3.6.1.4.1.9839.2.2.1.2.0
Digital	....	...
Digital	206	1.3.6.1.4.1.9839.2.2.1.206.0
Digital	207	1.3.6.1.4.1.9839.2.2.1.207.0
Analogue	1	1.3.6.1.4.1.9839.2.2.2.1.0
Analogue	2	1.3.6.1.4.1.9839.2.2.2.2.0
Analogue	...	...
Analogue	206	1.3.6.1.4.1.9839.2.2.2.206.0
Analogue	207	1.3.6.1.4.1.9839.2.2.2.207.0
Integer	1	1.3.6.1.4.1.9839.2.2.3.1.0
Integer	2	1.3.6.1.4.1.9839.2.2.3.2.0
Integer	...	...
Integer	206	1.3.6.1.4.1.9839.2.2.3.206.0
Integer	207	1.3.6.1.4.1.9839.2.2.3.207.0

Tab. 8.4.5.3

Following, a complete example of MIB tree, supposing a single unit (IR32Cold), with address 1, connected to the WebGate.

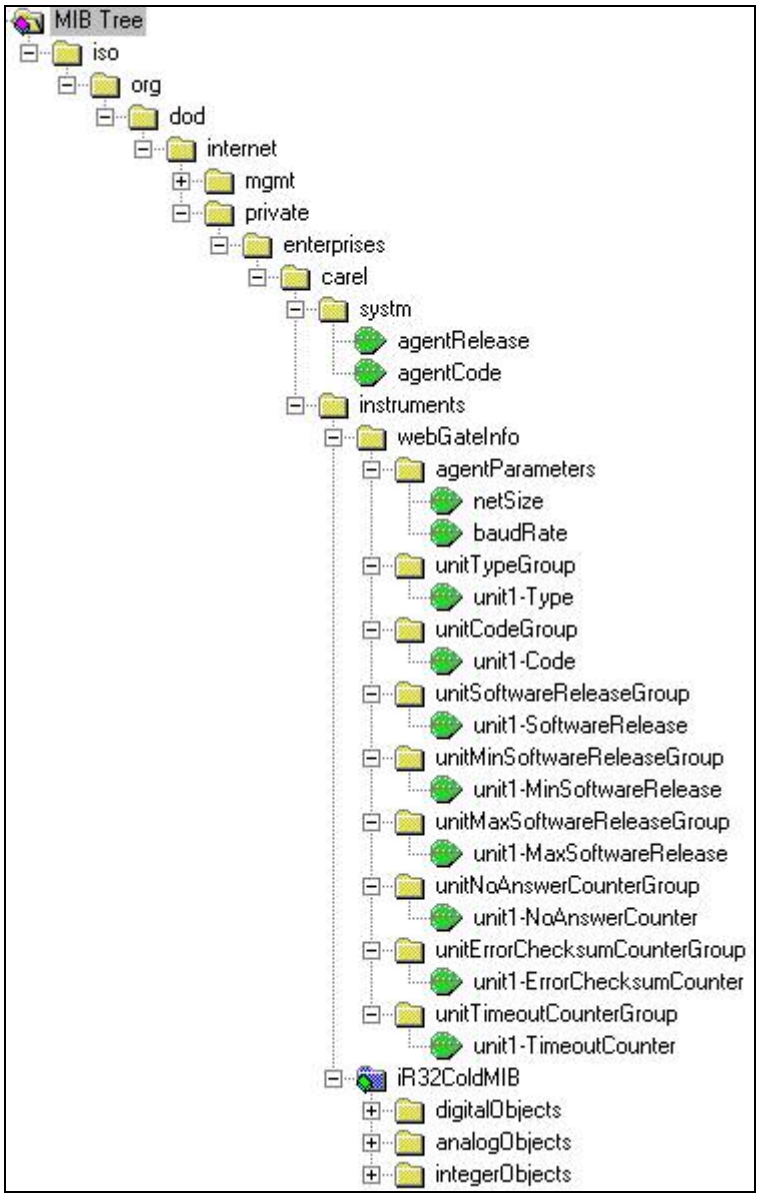


Fig. 8.4.5.2

8.5 SNMP command and version

The agent running on WebGate implements SNMPv.1 protocol. Consequently the handled message types are:

SNMPv.1
Message type
Get-request
GetNext-request
Get-response
Set-request

Tab. 8.5.1

8.6 Communities

The information in the protocol SNMP is based on the use of community names. Each SNMP message includes a community name that works like a password for communication between the manager and the agent. For the sintax of community names see the related functions in **WebGate Script Functions**.
WebGate has three community names:
READ-ONLY community: valid for *get* and *getnext* messagges.
READ-WRITE community: valid for all messages, including set operation.
TRAP community: valid for *TRAP* messages sent by the agent.

The default value for all community strings is “public”. When the user changes a community string, the new community name is saved in the configuration file, and it will be valid until the next change.
If the community name in the message sent by the manager is not valid, no response will be sent back by WebGate.

8.7 System MIB-II variables

MIB = Management Information Base is the main informations database for an SNMP network application. One of the most important groups under MIB-II node, is the *system group* (see Fig. 8.7.1). The three variables *sysContact*, *sysName* and *sysLocation* belonging to the system group are mandatory for any agent.

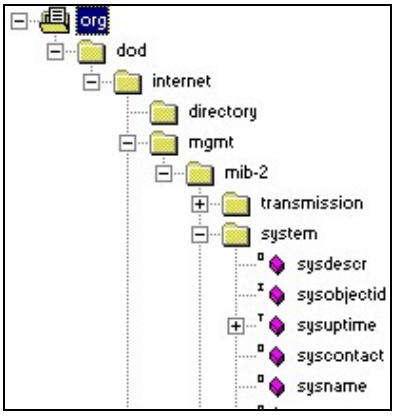
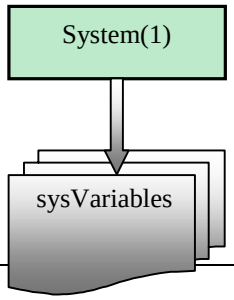


Fig. 8.7.1

System ::= iso(1).org(3).dod(6).internet(1).mgmt(2).mib-2(1).1



SysVariables = sysDescr ::= {system 1}
sysObjectID ::= {system 2}
sysUpTime ::= {system 3}
sysContact ::= {system 4}
sysName ::= {system 5}
sysLocation ::= {system 6}
sysServices ::= {system 7}

Fig. 8.7.2

The definition of the 3 MIB-II system variables managed by the WebGate is:

sysContact OBJECT-TYPE
SYNTAX OCTET STRING
ACCESS read-write
Status ---
DESCRIPTION “A person responsible for the node, along with information such as phone number”
::={System 4}

sysName OBJECT-TYPE
SYNTAX OCTET STRING
ACCESS read-write
Status ---
DESCRIPTION “Agent name”
::={System 5}

sysLocation OBJECT-TYPE
SYNTAX OCTET STRING
ACCESS read-write
Status ---
DESCRIPTION “The physical location of the device”
::={System 6}

WARNING
The maximum SNMP size for each string is:

Max size for WebGate system variables	
System variable	**Max size**
sysName	39 characters
sysContact, sysLocation	79 characters

Tab. 8.7.1

In addition, the string must be a string composed from alphanumeric characters *only* (0…9, A…Z and a…z). No spacing or punctuation characters are allowed.

8.8 TRAP messages

Trap messages enable an agent to report a serious condition or an important event to a manager station. The SNMP philosophy affirms that traps should be used carefully and sparingly. To respect that concept, the only SNMP trap message sent by WebGate is ***warmStart*** (1), signaling that the sender is reinitialising after a reboot, and its configuration will not change until next reboot.

8.8.1 TRAP destination

If no IP-Address has been ever set up as trap destination on the WebGate, the default destination is:

Default TRAP destination = 255.255.255.255 (disabled)

It means that no trap messages will be sent by WebGate, until a different destination is defined by the user.

8.9 Error Messages

The main SNMPv.1 error messages are:

SNMPv.1 Error messages	
Error type	**Description**
tooBig (1)	**The get-response message containing the result of an operation is bigger than the local implementation can handle.**
noSuchName (2)	One of the variables indicated in the request does not match anything in the relevant MIB view.
badValue (3)	A set-request asked the agent to write an inappropriate value, for example to write an integer when a text string was required.
readOnly (4)	A set-request tried to write a variable that the operator is not allowed to write.
genErr (5)	A variable can not be retrieved for some reason other than those listed above.

Tab. 8.9.1

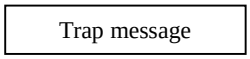
As we can see in table 8.9.1, error message sent by an agent depends on the manager request type.
Concerning WebGate we have the following cases:

WebGate Error Messages		
Error Messages	**PDU type***	**Possible cause and solution**
tooBig	G-GN	The length of the response message is too big. Try a new request querying for a number of variables less than the previous one.
noSuchName	G-S GN	The manager is querying for one or more variables that doesn't match anything in the MIBView of the WebGate. The error index indicates which is the first variables that caused the error. Check the OID of the variable and retry. Particular cases are querying for: • MIB-II system variables different by sysContact, sysName, sysLocation; • Objects not present in Carel-system subtree or under the Agent-parameters node; • For a General-Units-Info subtree with node index > 9 or, in such subtrees, for units having address > Netsize: • Unit with address > Netsize in the Carel-Instruments subtree. • The Index of a unit variable is greater than MaxIndex=207 in the Carel-Units-Var subtrees. Generally, when NMS asks for an object immediately successive at last in any subtree, WebGate sends this error message. Along with the general case, with correct OID, some particular items are querying for: • MIB-II system variables having OID-name less than sysUpTime or greater than sysName (the only system variables defined are sysContact, sysName, sysLocation); • Variable different by Agent-release in the Carel-system subtree; • Variable different by Netsize in Agent-parameters node (Fig.8); • Unit variable with index greater than MaxIndex-1.
readOnly	S	A set-request tried to write a variable that the operator is not allowed to write.
badValue	S	Particular cases are: • Writing not alphanumeric strings or too long for system variables; • Trying to set Netsize greater than 16 or wrong baudrate values;
genErr	G-GN-S	WebGate generally sends this error message when: • NMS tries to read software-releases of High-level units; • NMS tries operations on units that are off-line; • NMS tries a Set-request to write a unit variable and output-message-queue is full;

Tab. 8.9.2

(*): G=Get-request, GN=GetNext-request, S=Set-request;

Fig. 13



8.10 MTU dimension for WebGate SNMP

The Ethernet MTU (Maximum Transmission Unit) for WebGate is set to **576 bytes**. Do not send packets greater than this value, since they will be trashed and not recognized by the WebGate.

9. User Management

WebGate is provided with some basic user management capabilities.
User Management provides access restrictions when using functions and accessing to HTML pages and file system.

WebGate provides 4 basic access levels: "*Administrator*" (maximum access capabilities), "*Supervisor*", "*User*" and "*Guest*" (maximum restrictions).

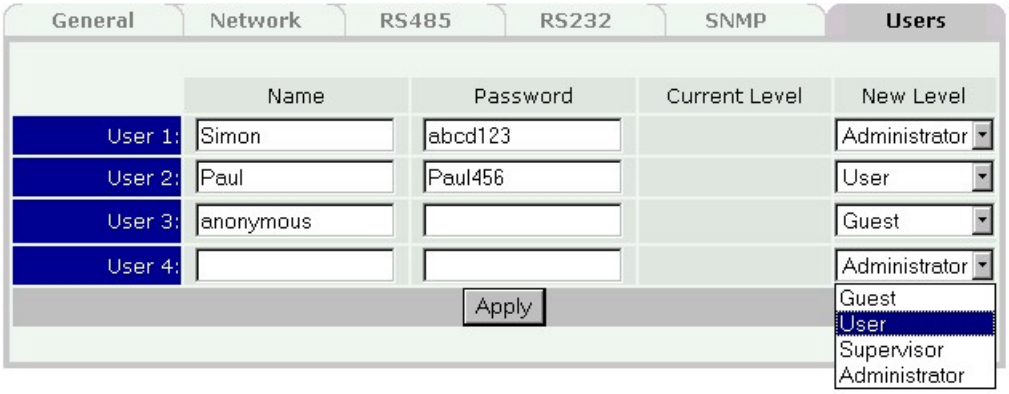
9.1 Access Restrictions

User Managements works in the following ways:

- *applying access restrictions to script functions.*
  Functions read access is allowed to everybody, but write access is limited to some users on a function-by-function basis, as indicated in chapter **WebGate Script Functions**.
  For instance, the "`Var`" function write access is limited to "Supervisor" and "Administrator" to avoid that everybody could change the settings of devices connected to the RS485 network. The association between a function and the minimum access level required is fixed and not modifiable.
- *Limiting access to the file system.*
  Read and write restrictions can be applied to any file, valid either using HTTP and FTP.
  File access levels are set through FTP. Please see chapter **File Transfer Protocol** for an extensive description of FTP.

9.2 Users Definition

WebGate user management is based on a "user table" that associates every user with a password and an access level. The simplest way to set user properties is through the "Users" tab in the predefined WebGate configuration page, as indicated below:



9.3 Naming Conventions

User names and passwords must be composed of at most 20 alphanumeric characters, without spacing.
Furthermore, names and passwords are *case sensitive:* this means that "Paul" and "paul" is *not* the same user.
If you want to remove an user from the table, simply clear his name from the user table.
If the "password" field is left empty, the associated user will gain access with any password.

9.4 "anonymous" User

Anonymous access is used when somebody connects for the first time to WebGate with a browser, a FTP client or through the console.The anonymous user is the one with name field set to "`anonymous`".
By default, if the anonymous user doesn't appear in the user table, WebGate assign to him maximum access capabilities ("`Administrator`"). Consequently, if you don't need user management you can simply ignore the user management capabilities provided by the WebGate. However, to proficiently use *user management* capabilities you will have to create an anonymous user in the table, assigning to him a proper access level, and/or a password.
For example, refer to the previous image where to the "anonymous" user ("User 3") was assigned the "`guest`" access, without any password.
WARNING. Take care to have an Administrator user before changing the "anonymous" access level from "Administrator" to a lower level

9.5 "`guest`" access level and passwords

Password are not required for "`guest`" access level when accessing to HTML pages. We RECOMMEND NOT *to set a password for guest users, because not supported*. Please note that as a final result, *any file placed in the WebGate root directory is always visible without any password*.

9.6 Access Levels for Factory Shipped Pages

HTML pages shipped with the WebGate for introduction ("index.html") and general information ("wg_info.htm") are readable from any user. In contrast, all configuration pages requires "`Administrator`" level.

9.7 Accessing to Protected HTML Pages

HTML protection is provided through the use of the browser password management capabilities. When you try to view a protected page, the browser will display a password request window similar to the one depicted on the right in fig.9.7.1:

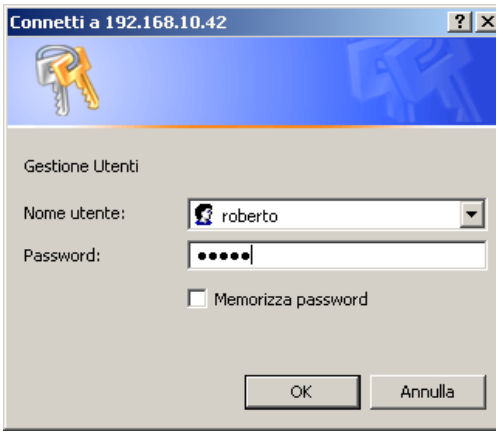


Fig. 9.7.1

If a valid combination of name and password is not provided, the access is forbidden.

WARNING: As said before, WebGate configuration pages requires at least "`Administrator`" access level. Since of this, be careful when creating users, because first of all you must create at least an Administrator. If you fail to create an administrator or if you forget your administrator password, *you will have to completely erase WebGate contents*, following the procedure described in *Total Erase*.

Please note that when creating the first administrator the password request window will be visualized immediately after pressing the "apply" button in the "Users" configuration tab.

9.8 Security Issues

WebGate user management is provided as a basic method to discourage unauthorized access but is NOT a security feature. No encoded protocols or particular protections against security attacks are implemented. Since of this, any security improvement required from the customer must be assured at network level with firewalls and similar devices.

9.8.1 POST and GET methods

When creating an HTML page (see **Creating a custom Web page on the WebGate**), if the page requested by the "action" attribute has a protection level greater than "`guest`" it is recommended to use the POST method instead of GET to submit data, because GET method allows you to show only pagewith "`guest`" access level.

9.9 Advanced User Table Management

Through the use of "`UserLevel`", "`UserName`" and "`UserPwd`" functions user magament can be further improved, in fact:

- ✓ Despite only 4 users are displayed in the standard user tab, up to 6 table entries can be created.
- ✓ User access levels can be subdivided in more sublevels than the four indicated (however, the file system directory tree is fixed, as indicated in (**Directories and "Read Access" file protection**).

Basically, an access level among a range of values is associated to each class (see Tab. 9.9.1). By default the access level associated to each class is the higher value in the range: 99 for "`guest`", 149 for "`user`", 199 for "`supervisor`" and 249 for "`administrator`".

Membership Class	Access-Level
`Guest`	From 0 to 99
`User`	From 100 to 149
`Supervisor`	From 150 to 199
`Administrator`	From 200 to 249

Tab. 9.9.1

For a description of "`UserLevel`", "`UserName`" and "`UserPwd`" functions please see WebGate Script Functions.
The customer can build his own page to be able to set the access-level he likes (from 0 to 249).
Fig. 9.9.1 is an example of how a this page can be built.

Fig. 9.9.1

10. Firmware Upgrade

WebGate ensures that upgrades can be easily obtain since the firmware is stored into an embedded rewritable memory.
The upgrade can be performed remotely copying a file supplied by Carel into the file system root directory using a FTP client program and rebooting the device.
Upgrade files have a "*.pack*" extension and can update not only the firmware, but also factory HTML pages and the low-level devices configuration table used to communicate with some Carel controls.

•

To proceed for an upgrade follow the following instructions:

1. The upgrade file needs some space to be stored (typically about 300Kbyte), and it may also require some additional space during and after the installation. Since of this, as a general rule, remove any user installed file.
2. Copy the file supplied by Carel into the WebGate root directory using a FTP client program. To do this you don't need any special access right.
3. Open with a browser the "*General*" configuration page , check the "reboot" checkbox and press "Apply".
4. The update operation now proceed automatically and may take some minutes.
5. Open the "*index.html*" page. If everything goes fine, you should see the new firmware release (if changed from the update) and in the "last update status" line you should read "*Update file not found*". This is normal since the update was correctly performed and the file you copied at step 2 has been removed from WebGate.

Note
If the file space is not sufficient to complete the upgrade, or if any other error occurred, after the reboot a message in the "last update status" line of the HTML "*information*" page will address the problem.

IMPORTANT WARNING.
Absolutely never remove power supply when the WebGate is performing a firmware update, since this could severely damage the device and require technical assistance.

11. WebGate Script Functions

The WebGate console and HTML user interface is based on some script “functions”.
Script Functions are provided principally to allow the user to read from and write to Carel network devices and to change specific WebGate configuration parameters (for example its IP Address)), and can be processed if present in the file with extension “**.htm**”, “**.html**” or “**.js**”.
A function may be parametric or without any parameter (“command”). Function names are not case sensitive.

In this chapter all the available functions are listed and individually described.

11.1 Functions in alphabetical order:

AccessLevel	**AccessLevelString**	**Baudrate232**
Baudrate485	**ClearConfig**	**Ctrl232**
DebugStrings	**Dev**	**DevInfo**
Eval	**FlashFormat**	**FreeFiles**
Gateway	**Help**	**HWVersion**
IPAddress	**IsSelBaudrate232**	**IsSelBaudrate485**
IsSelUserLevel	**Login**	**Logout**
LumpTest	**MACAddress**	**MaxDevs**
NetMask	**NetStat**	**Option**
Passwd	**Reboot**	**Registers**
ROCommunity	**RWCommunity**	**ScanDevs**
Set	**SetMax**	**SetMin**
SNMPAgentRel	**SNMPSyscont**	**SNMPSysloc**
SNMPSysname	**Status**	**Status485**
SWVersion	**TrapCommunity**	**TrapIPAddress**
UpdateStatus	**UserLevel**	**UserLevelString**
UserName	**UserPwd**	**Var**
WakeOn		

11.2 Functions sorted by category:

Console	
Login	Login a new user to the console
Logout	Logout the current user from the console
Passwd	Enter the user password for the console
RS232	
Baudrate232	Get/set the baudrate for the RS232 serial interface
Ctrl232	Give access to specific RS232 control lines
IsSelBaudrate232	Return a specific string when the current baudrate match the given parameter
RS485 (Carel Network)	
Baudrate485	Get/set the baudrate for the RS485 serial interface
Dev	Display concise information about a device
DevInfo	Display extended information about a device
IsSelBaudrate485	Return a specific string when the current baudrate match the given parameter
MaxDevs	Set the maximum number of devices connected to the WebGate
ScanDevs	Rescan all the devices connected to WebGate
Status485	Get information about the RS485 and Carel Network interface
Var	Get/set a device variable value
Ethernet – LAN	
Gateway	Get/set the network gateway IP Address
IPAddress	Get/set the WebGate IP Address
MACAddress	Return the WebGate Ethernet hardware address (MAC)
NetMask	Get/set the network mask pattern
NetStat	Displays some network statistics
SNMP	
ROCommunity	Get/set the Read Only Community Name
RWCommunity	Get/set the Read/Write Community Name
SNMPAgentRel	Return the Agent software release
SNMPSyscont	Get/set the System Contact
SNMPSysloc	Get/set the System Location
SNMPSysname	Get/set the System Name
TrapCommunity	Get/set the Trap Community Name
TrapIPAddress	Get/set the trap manager IP Address
• **User Management**	
AccessLevel	Return the current user access level
AccessLevelString	Return the minimum access level descriptive string for the current user
IsSelUserLevel	Return a specific string when the current user level match the given parameter
UserLevel	Get/set a user access level
UserLevelString	Return the minimum access level descriptive string for a given user
UserName	Get/set a user name
UserPwd	Get/set a user password
Other	
Eval	Evaluate a numerical expression
FreeFiles	Return the amount of memory free for files storage
Help	Show a list of all the available functions
HWVersion	Return the WebGate hardware version
Reboot	Restart the WebGate
Registers	Displays a list of all the currently defined registers
Set	Assigns a numeric value to a register
SetMin	Set the minimum allowed value for a register
SetMax	Set the maximum allowed value for a register
Status	Displays the current WebGate status
SWVersion	Return the WebGate software version
UpdateStatus	Return the last reboot update error code
WakeOn	Displays the time elapsed since the last WebGate reboot
Reserved	
ClearConfig	This command is reserved and must not be used
DebugStrings	This command is reserved and must not be used
FlashFormat	This command is reserved and must not be used
LumpTest	This command is reserved and must not be used
Option	This command is reserved and must not be used

11.3 Detailed commands description:

In the following paragraphs all commands are described in detail, in alphabetical order.

In the upper title, the description items have the following meanings:

- "***Read***" indicates whether the command can return a value when read (✓) or not (✗),
- "**Write**" is the minimum user access level required for write access (when allowed). Please note that read access is never restricted by user management system instead.
- "**Saved**" indicates whether the command performed is "saved" (✓) in the WebGate persistent configuration memory and reloaded at every reboot, or not (✗).

Every paragraph is completed with a simple console example, since this interface is the most straightforward to experiment with. However, any function can be used, without restrictions, in HTML pages too. In this last case, please note that some commands don't make any output value visible in the browser or, for some errors, a dedicated "warning page" to be displayed can be generated.

Also if not indicated for each function, when a function name is misspelled or values are outside the allowed ranges, generally an error is returned.

As a final notice, please note that the expression "user" indicated in the following paragraphs refers to any operator currently accessing the WebGate and is not referred to the "`user`" *level of access*, if not expressly written .

AccessLevel	•		
Description	*Read*	*Write*	*Saved*
Return the current user access level	✓	✗	✗

Displays the current user access level in numeric format.
This function is available either for console or HTML.
Please see chapter **User Management** for a description of user management system.

Syntax:
`AccessLevel`

Console example:

```
> AccessLevel↵
249

> _
```

AccessLevelString	•		
Description	*Read*	*Write*	*Saved*
Return the minimum access level descriptive string for the current user	✓	✗	✗

Displays the current user access level in textual format.
Please see chapter **User Management** for a description of user management system.

Syntax:
`AccessLevelString`

Returned Value:
The returned value is one of the following:

`Guest`	(0 ≤ user access level ≤ 99)
`User`	(100 ≤ user access level ≤ 149)
`Supervisor`	(150 ≤ user access level ≤ 199)
`Administrator`	(200 ≤ user access level ≤ 249)

Console example:

```
> AccessLevelString↵
administrator

> _
```

Baudrate232	•		
Description	*Read*	*Write*	*Saved*
Get/set the baudrate for the RS232 serial interface	✓	**Administrator**	✓

This function displays or changes the current RS232 baudrate. Currently, this value is used by the console user interface to communicate with a terminal emulator.

Syntax (*read*):
`Baudrate232`

Syntax (*write*):
`Baudrate232 = <value>`

Allowed values:
1200, 2400, 4800, 9600, 19200, 38400

If an invalid value is entered, an error string will be returned.

Default Value:
19200 Baud

Console example:

```
> Baudrate232=12 ↵        ← Wrong !
Invalid baudrate (code -401)

> Baudrate232=19200 ↵

> Baudrate232 ↵
19200

> _
```

Baudrate485	•		
Description	*Read*	*Write*	*Saved*
Get/set the baudrate for the RS485 serial interface	✓	**Administrator**	✓

This function displays or changes the current RS485 baudrate. This value is used by the Carel Network interface to communicate with Carel devices.

Syntax (*read*):
`Baudrate485`

Syntax (*write*):
`Baudrate485 = <value>`

Allowed values:
1200, 2400, 4800, 9600, 19200

If an invalid value is entered, an error string will be returned.

Default Value:
19200 Baud

Console example:

```
> Baudrate485=12 ↵              ← Wrong !
Invalid baudrate (code -401)

> Baudrate485=19200 ↵

> Baudrate485 ↵
19200

> _
```

Ctrl232	•		
Description	*Read*	*Write*	*Saved*
Give access to specific RS232 control lines	✓	×	×

***NOTE:** This function is provided for debug purposes only. Write access is always forbidden.*

Get the value of RS232 status and control lines.

Syntax:
`Ctrl232`

Returned Value:
It returns a hexadecimal value, with the bit meanings indicated below:

7	6	5	4	3	2	1	0
CD		DSR	CTS			RTS	DTR

Every bit set to "1" indicates that the corresponding line is active.

In the Chapter **Connection of the RS232** interface you can find an extended description of any bit and its correspondence with serial connector pins.

Console example:

```
> Ctrl232↵
0xb3

> _
```

Dev	•		
Description	*Read*	*Write*	*Saved*
Display concise information about a device	✓	×	×

Returns the status and type code of a connected Carel device.

Syntax:
`Dev(<address>)`

`<address>` **represents the physical address of the unit of interest.**
If this value if less than '1' or greater than the value returned from the "MaxDevs" function, WebGate returns a warning message.

Returned Value :
The "type code" is a number identifying the type of device connected. In the table below are reported for reference some device codes. Since new devices are continuosly developed, you are advised that this may not be an exhaustive list.

Code	Description
1, 2, 3	Asynchronous I/F board for IR32/µChiller (indicating firmware release)
10	µChiller compact
21	MPX
101	pCO
102	I/O pCO² Expansion (pCo with custom BIOS)
110	MGE MPX
111…119	Reserved for MGE
120	Power split
121	Power split cell
148	IR plug-in
149	mP30 Uniflair
150	FCM
151	µAC Carel
152	Humidifier controller URC (resistors)
153	Humidifier controller UEC (electrodes)
154	Humidifier controller UEC (electrodes, low cost)
155	Humidifier controller (gas)
156	Humidifier controller (atomization)
200	PlantWatch Printer (unit address 21h, reserved 21h-25h)
201	pCO ²
202	PST Terminal (reserved address 26h-29h)

If the required device is configured (that is: it *was* recognized from Webgate) but it is currently off-line, WebGate returns the type code of the last device with the given address.

Console example:

```
> Dev(1)↵
Low-Level Unit ON-LINE: Type code 10

> _
```

DevInfo	•		
Description	*Read*	*Write*	*Saved*
Get extended information about a device	✓	×	×

Returns a complete status report of a connected Carel device.

Syntax:
`DevInfo(<address>)`

`<address>` **represents the physical address of the unit of interest.**
If this value if less than '1' or greater than the value returned from the "MaxDevs" function, WebGate returns a warning message.

Returned Value :
The information returned can be useful to identify a device or to debug network conections that seems to be poor. Please refer to "***Dev***" function for a description of "type codes"

Console example:

```
> DevInfo(1)↵
Low-Level Unit ON-LINE: Type code 10
Forcing by master to read all variables completed.
Software Release 13
No answer counter 0
CheckSum error counter 0
Timeout counter 0
Software release min.: 13
Software release max.: 13

> _
```

Eval	•		
Description	*Read*	*Write*	*Saved*
Evaluate a numerical expression	✓	×	×

This function evaluates an arithmetic expression and returns the result. It may prove useful when creating HTML pages.

Syntax:
`Eval(<expression>)`

`<expression>` Arithmetic expressions are explained in **About Expressions, Registers and Functions**: they may be numbers, registers, or a combination of the two

Returned Value :
Numerical result of the operations implied in the expression.

Console example:

```
> Set(Address)=7↵

> Eval(Address – 1)↵
6

> Set(Index)=Address+2↵ ← DO NOT insert blanks here

> Eval(Index – Address + 1)↵
3

> _
```

FreeFiles	•		
Description	*Read*	*Write*	*Saved*
Return the amount of memory free for files storage	✓	✖	✖

Return how many bytes and file locations are free to store files in the WebGate file system.
Files can be written to the WebGate using a PC running a FTP client application.

Syntax:
`FreeFiles`

Returned Value :
The value returned is a string like the following:
`<f>` **`free of`** ***`<t>`*** **`(comprising`** ***`<mf>`*** **`mirrored of`** ***`<mt>`*** **`). `*`<fa>`*** **`files available`**

The fields highlighted with <> have the following meanings:
<f> : number of bytes free to the user for file storage
<t> : number of total bytes available in the file system[1]
<mf> : number of bytes free in the mirrored part of the file system[2]
<mt> : number of total bytes available in the mirrored part of the file system[2]
<fa> : number of file locations already free

Space available to user:
WebGate WEBG0000B0 provides the user with about 400KByte of memory available for files storage. A maximum of 100 files can be stored at the same time.

Note about the space occupied by files:
Files are stored with a "header" of about 150 bytes in sectors of 1KByte each. Consequently, every file will use a memory area a little greater than the size of the file itself.

Console example:

```
> FreeFiles↵
15360 free of 516096 (comprising 25600 mirrored of 129024 ). 98 files available.

> _
```

Gateway	•		
Description	*Read*	*Write*	*Saved*
Get/set the network gateway IP address	✓	**Administrator**	✓

Displays or changes the network gateway address. It allows to set the IP address of an Ethernet gateway which is used to forward IP packets to a destination not directly attached to the same subnet defined by the combination of "IPAddress" and "NetMask".

Syntax (*read*):
`Gateway`

Returned value for read:
`<IPx>.<IPy>.<IPz>.<IPw>`

Where *`<IPx>`*, *`<IPy>`*, *`<IPz>`* and *`<IPw>`* are the four bytes of IP expressed using the standard dotted format string.

Syntax (*write*):
`Gateway = `*`<IPx>.<IPy>.<IPz>.<IPw>`*

Where *`<IPx>`*, *`<IPy>`*, *`<IPz>`* and *`<IPw>`* are the four bytes of IP expressed using the standard dotted decimal format string.
Please contact your system administrator to obtain the proper gateway address. If you don't need to use a gateway, disable it using the following commands: **`Gateway = 255.255.255.255`**,
and **`NetMask = 0.0.0.0`**

Allowed values:
Any combination of four numbers in the range 0…255

[1] A small amount of this area (tipically about 100 Kbytes) is reserved and cannot be freed by the user.
[2] This memory is reserved and is not available to the users

Note:
When you change gateway address, WebGate will immediately store the value just selected. However, the old value will be used until a connection (FTP, HTTP or SNMP) is still active. As a consequence, any tentative to obtain a connection through the new gateway during this time interval will fail. To avoid this and to perform the operation faster, we suggest to change it, if possible, when no other user's requests are active.

Returned value for write:
One of the following situations may occur:

Valid value entered and no connections still active:	`IP value updated!`
Valid value entered and one or more connections still active:	`Valid for next connection!`
Invalid value entered:	"bad value" or "syntax error" string.

Default Value:
255.255.255.255 (disabled)

Console example:

```
> Gateway↵
192.168.0.32

> Gateway=192.168.0.20↵
IP value updated!

> _
```

Help •

Description	*Read*	*Write*	*Saved*
Show a list of all the available functions	✓	×	×

This command gets a list of all the commands and parameters available for use by console interface or HTML.
The characters following the function name indicates what the user is allowed to do with that function:

- `R` **: means that the user is allowed to use that function for read**
- **`W`** **:** means that the user is allowed to use that function for write
- **`-C`** **:** means that the item is a command (that is, it requires neither parameters nor value)
- **`--`** **:** means that the function is not available to the current user.

Please note that some functions are not allowed for read or write at all.
Moreover, the availability of each function is dependant also on the current user access level (for example, the "*var*" function is allowed only to supervisors and administrators).

Syntax:
Help

Console example:

```
> Help↵
Available commands and variables are:
AccessLevel         R- |  Option             RW
AccessLevelString   R- |  Passwd             -W
Baudrate232         RW |  Reboot             -C
Baudrate485         RW |  Registers          R-
ClearConfig         -C |  ROCommunity        RW
Ctrl232             RW |  RWCommunity        RW
DebugStrings        RW |  ScanDevs           -C
Dev                 R- |  Set                -W
DevInfo             R- |  SetMax             -W
Eval                R- |  SetMin             -W
FlashFormat         -C |  SNMPAgentRel       R-
FreeFiles           R- |  SNMPSyscont        RW
Gateway             RW |  SNMPSysloc         RW
Help                R- |  SNMPSysname        RW
HWVersion           R- |  Status             R-
IPAddress           RW |  Status485          R-
IsSelBaudrate232    R- |  SWVersion          R-
IsSelBaudrate485    R- |  TRAPCommunity      RW
```

```
IsSelUserLevel     R- |  TRAPIPAddress      RW
Login              -W |  UpdateStatus       R-
Logout             -C |  UserLevel          RW
LumpTest           RW |  UserLevelString    R-
MACAddress         RW |  UserName           RW
MaxDevs            RW |  UserPwd            RW
NetMask            RW |  Var                RW
NetStat            R- |  WakeOn             R-

> _
```

HWVersion	•		
Description	*Read*	*Write*	*Saved*
Return the WebGate hardware version	✓	✖	✖

This function returns the WebGate hardware version (revision).

Syntax:
HWVersion

Returned value:
***<Version>*.*<Release>* (buid *<build>*)**

The above format was choosen to be compatible with the "*SWVersion*" command.

<Version> **is used to identify a major hardware change**
<Release> **is used to identify a minor hardware change**
<Build> **currently is not used. Its value is fixed to zero.**

Please note that the value returned from this function is only indicative and not related to the "true" hardware version reported on the label attached to the WebGate. When contacting Carel for support, always report the release and serial number indicated on the label.

Console example:

```
> HWVersion↵
(build 0)

> _
```

IPAddress	•		
Description	*Read*	*Write*	*Saved*
Get/set the WebGate IP address	✓	**Administrator**	✓

Displays or changes the IP address of the WebGate itself.

Syntax (*read*):
IPAddress

Returned value for read:
<IPx>*.*<IPy>*.*<IPz>*.*<IPw>

Where *<IPx>*, *<IPy>*, *<IPz>* and *<IPw>* are the four bytes of IP expressed using the standard dotted format string.

Syntax (*write*):
IPAddress = <IPx>.<IPy>.<IPz>.<IPw>

Where <IPx>, <IPy>, <IPz> and <IPw> are the four bytes of IP expressed using the standard dotted decimal format string. Please contact your system administrator to obtain a valid IP address. Anyway, remember that every network appliance *must* have its own IP address. Carefully avoid to use the same IP address for more appliances. Conflicting addresses is a common configuration pitfall that will result in network malfunctions.

Allowed values:
Any combination of four numbers in the range 0…255

Note:
When you change IP address, WebGate will immediately store the value just selected. However, the old value will be used until a connection (FTP, HTTP or SNMP) is still active. As a consequence, any tentative to obtain a connection using the new IP value during this time interval will fail. To avoid this and to perform the operation faster, we suggest to change it, if possible, when no other user's requests are active.

Returned value for write:
One of the following situations may occur:

Valid value entered and no connections still active:	`IP value updated!`
Valid value entered and one or more connections still active:	`Valid for next connection!`
Invalid value entered:	"bad value" or "syntax error" string.

Default Value:
192.168.0.250

Console example:

```
> IPAddress=192.168.0.200↵
IP value updated!

> IPAddress↵
192.168.0.200

> _
```

IsSelBaudrate232 •

Description	*Read*	*Write*	*Saved*
Return a specific string when the current baudrate match the given parameter	✓	×	×

This function returns a value depending on the match of the given parameter with the current RS232 baudrate value. It is useful in combination with drop-down selection lists.

Syntax:
`IsSelBaudrate232(`*`<expression>`*`)`

`<expression>` **represents the baudrate to compare with.**

Returned value:
Empty string if the baudrate doesn't match, or
`selected` **if the baudrate is the same.**

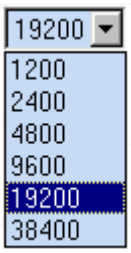
Console example:

```
> Baudrate232=19200↵

> IsSelBaudrate232(2400)↵

> IsSelBaudrate232(19200)↵
 Selected

> _
```

HTML example:

```
<select name="select3" size="1">
<option value="?script:baudrate232=1200"
<%IsSelBaudrate232(1200)%>>1200</option>
<option value="?script:baudrate232=2400"
<%IsSelBaudrate232(2400)%>>2400</option>
<option value="?script:baudrate232=4800"
<%IsSelBaudrate232(4800)%>>4800</option>
<option value="?script:baudrate232=9600"
<%IsSelBaudrate232(9600)%>>9600</option>
<option value="?script:baudrate232=19200"
<%IsSelBaudrate232(19200)%>>19200</option>
<option value="?script:baudrate232=38400"
<%IsSelBaudrate232(38400)%>>38400</option>
</select>
```

Graphical result:



IsSelBaudrate485 •

Description	*Read*	*Write*	*Saved*
Return a specific string when the current baudrate match the given parameter	✓	×	×

This function returns a value depending on the match of the given parameter with the current RS485 baudrate value. It is useful in combination with drop-down selection lists.

Syntax:
`IsSelBaudrate485(`*`<expression>`*`)`

`<expression>` **represents the baudrate to compare with.**

Returned value:
Empty string if the baudrate doesn't match, or
`selected` **if the baudrate is the same.**

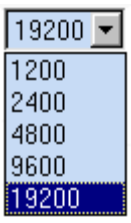
Console example:

```
> Baudrate485=19200 ↵

> IsSelBaudrate485(2400) ↵

> IsSelBaudrate485(19200) ↵
 Selected

> _
```

HTML example:

```
<select name="select3" size="1">
<option value="?script:baudrate485=1200"
<%IsSelBaudrate485(1200)%>>1200</option>
<option value="?script:baudrate485=2400"
<%IsSelBaudrate485(2400)%>>2400</option>
<option value="?script:baudrate485=4800"
<%IsSelBaudrate485(4800)%>>4800</option>
<option value="?script:baudrate485=9600"
<%IsSelBaudrate485(9600)%>>9600</option>
<option value="?script:baudrate485=19200"
<%IsSelBaudrate485(19200)%>>19200</option>
</select>
```

Graphical result:



IsSelUserLevel	•		
Description	*Read*	*Write*	*Saved*
Return a specific string when the current user level match the given parameter	✓	×	×

This function returns a value depending on the match of the given user and access level with the current value. It is useful in combination with drop-down selection lists.
Please see chapter **User Management** for a description of user management system.

Syntax:
`IsSelUserLevel(<index>, <level>)`

`<index>` **represents the user index in the users table (0…5)**
`<level>` **represents the user level to compare with**

Returned value:
Empty string if the access level doesn't match, or
`selected` **if the access level is the same.**

Console example:

```
> UserLevel(1) = 200↵

> IsSelUserLevel(1, 100)↵

> IsSelUserLevel (1, 200)↵
 Selected

> _
```

HTML example:

```
<select name="select">
<option value="?script:UserLevel(1)=99"
<%IsSelUserLevel(1, 99)%>>Guest</option>
<option value="?script:UserLevel(1)=149"
<%IsSelUserLevel(1, 149)%>>User</option>
<option value="?script:UserLevel(1)=199"
<%IsSelUserLevel(1,
199)%>>Supervisor</option>
<option value="?script:UserLevel(1)=249"
<%IsSelUserLevel(1,
249)%>>Administrator</option>
</select>
```

Graphical result:

Login	•		
Description	*Read*	*Write*	*Saved*
Login a new user to the console	×	**Guest**	×

Give console access rights to an user.
The console interface fully comply with the restrictions imposed from the users management system, described in chapter **User Management**.
To gain access to WebGate through the console interface, a user is required to perform a two steps operation first:

1. Enter his/her name using the *Login* command
2. Enter his/her password using the *Passwd* command

When a wrong password is entered, an error is issued and the entire process must be repeated from the beginning.
Moreover, please note that the *login* command will not issue any immediate error if the entered name is not defined in the users table. The error will be addressed from the next *passwd* command.

Syntax:
Login = <*UserName*>
***Where* <*UserName*> represents the user name to enter with.**

Returned value:
"Already logged in, you must logout first (code -311)" if another user is already logged in
Error message if the string is not a valid name
No message if the string seems to be a valid name

Note:
When WebGate reboots, no user is automatically logged in. To prevent unauthorized access, users are automatically logged out after 5 minutes from the last console input. When no user is logged in, the console can be used anyway. However, the access is restricted to the same rights of the "anonymous" user (see "anonymous" User). By default the anonymous user have administrator rights. Consequently, if you don't need any security capability, you won't need to use "login" command.

Console example:

```
> Logout↵

> Login = JohnDoe↵

> Passwd =↵
Welcome JohnDoe! Your access level is 249 (administrator)

> _
```

Logout	•		
Description	*Read*	*Write*	*Saved*
Logout the current user from the console	×	**Guest**	×

This command logout the current user from the console, and return the WebGate to the default "anonymous" access level (see chapter **User Management**).
You must use this command each time you are logged in and you want to re-enter with another user name.

Syntax:
Logout

Returned value:
None

Note:
When WebGate reboots, no user is automatically logged in. To prevent unauthorized access, users are automatically logged out after 5 minutes from the last console input.
When no user is logged in, the console can be used anyway. However, the access is restricted to the same rights of the "anonymous" user (see "anonymous" User). By default the anonymous user have administrator rights. Consequently, if you don't need any security capability, you won't need to use "login" command.

Console example:

```
> Logout↵

> Login = JohnDoe↵

> Passwd =↵
Welcome JohnDoe! Your access level is 249 (administrator)

> _
```

MACAddress	•		
Description	*Read*	*Write*	*Saved*
Return the WebGate Ethernet hardware address (MAC)	✓	×	×

This function return the WebGate Ethernet hardware address.
Sometimes, this may be useful to configure or debug a network.

The MAC address is a worldwide unique number jointly assigned by IEEE and Carel. Consequently, MAC address is a permanently stored value that can't be changed from the user.

Syntax:
`Gateway`

Returned value:
`<MACa>:< MACb>:<MACc>:<MACd>:<MACe>:<MACf>`

Where *`<MACa>`*, *`< MACb>`*, *`<MACc>`*, *`<MACd>`*, *`<MACe>`* and *`<MACf>`* are the six bytes identifying the address expressed using hexadecimal octects.

Console example:

```
> MACAddress↵
00:90:c2:80:16:5a

> _
```

MaxDevs	•		
Description	*Read*	*Write*	*Saved*
Set the maximum number of devices connected to WebGate	✓	**Administrator**	×

This function is used to set (or obtain) the maximum number of devices that can be connected through the RS485 Carel interface.
Specifically, this value limits the address range of peripherals "scanned" from WebGate:
For example, if MaxDevs is set to "3", only devices with address "1", "2" and "3" will be scanned.
If this command is used to inform WebGate of the right number of devices connected, and the devices are sequentially numbered you will get the following benefits:

- Faster network scan
- The RS485 Led will indicate "red" only when a device is effectively disconnected

Syntax (*read*):
`MaxDevs`

Syntax (*write*):
`MaxDevs = <value>`

Allowed values:
1 up to 16

If an invalid value is entered, an error string will be returned.

Please note that when this value is changed, the network is immediately rescanned (see *`ScanDevs`* command)

Default Value:
16 (this is the maximum number of devices that can be interfaced).

Console example:

```
> MaxDevs=4↵

> MaxDevs↵
4

> _
```

NetMask			
Description	*Read*	*Write*	*Saved*
Get/set the network mask pattern	✓	**Administrator**	✓

Displays or changes the network mask pattern ("*netmask*").

Syntax (*read*):
`NetMask`

Returned value for read:
`<IPx>.<IPy>.<IPz>.<IPw>`

Where *`<IPx>`*, *`<IPy>`*, *`<IPz>`* and *`<IPw>`* are the four bytes of mask expressed using the standard IP dotted format string.

Syntax (*write*):
`NetMask = <IPx>.<IPy>.<IPz>.<IPw>`

Where *`<IPx>`*, *`<IPy>`*, *`<IPz>`* and *`<IPw>`* are the four bytes of mask expressed using the standard IP dotted decimal format string.
Please contact your system administrator to obtain the proper netmask. If you don't need to use a gateway, disable it using the following commands: **`Gateway = 255.255.255.255`**, and **`NetMask = 0.0.0.0`**

Allowed values:
Any combination of four numbers in the range 0…255

Note: When you change netmask, WebGate will immediately store the value just selected. However, the old value will be used until a connection (FTP, HTTP or SNMP) is still active. As a consequence, a tentative to obtain a connection using the new setting during this time interval will fail. To avoid this and to perform the operation faster, we suggest to change it, if possible, when no other user's requests are active.

Returned value for write:
One of the following situations may occur:

Valid value entered and no connections still active: `IP value updated!`

Valid value entered and one or more connections still active: `Valid for next connection!`

Invalid value entered: "bad value" or "syntax error" string.

Default Value:
0.0.0.0 (no gateway is used)

A concise note about IP routing
Although a comprensive discussion of networking issues are out of the scope of this document, we want to briefly add an additional comment about netmask and IP routing. Together with the "IP address", the netmask defines a sub-network ("subnet") where an appliance is, and instruct the appliance itself when a message must be redirected to a gateway. A subnet is a group of communicating appliances that are in direct connection between them. When a remote destination, external to the subnet, must be accessed from the host, the message must pass through a gateway that will redirect it to the right location ("IP routing").
The subnet is defined performing a bitwise and-mask between the IP address and the netmask.
When the value obtained "and-masking" a destination address with the netmask is the same value obtained "and-masking" the IP address of the sender with the netmask, the message is sent directly, otherwise it is redirected to the Gateway.
For example, if the sender IP address is "192.168.0.250" and the netmask is "255.255.0.0", any message for "192.168.X.X" will be send directly, but a message sent to "192.20.0.0" will pass through to gateway.

Console example:

```
> NetMask↵
192.168.0.0

> NetMask=0.0.0.0↵
IP value updated!

> _
```

NetStat	•		
Description	*Read*	*Write*	*Saved*
Display some network statistics	✓	✖	✖

This command gets a list of Ethernet network statistics.
This may be useful to advanced users or system administrators for debug purposes.

Syntax:
- **`Help`**

Returned Value:
- **The lines returned have the following meanings:**

Line	Meaning
`ARP Packets`	Number of ARP packets received[1]. ARP packets are broadcast messages used to “find” a remote host.
`TCP Packets`	Number of TCP packets received[1]. Generally point-to-point messages used for HTTP and FTP.
`UDP Packets`	Number of TCP packets received[1]. Generally point-to-point messages used for SNMP.
`ICMP Packets`	Number of ICMP packets received[1]. These packets are used for network management
`Other IP Packets`	Number of unhandled IP packets [1].
`Other Not IP Packets`	Number of unhandled packets other than IP [1].
`IP Fragments`	Number of fragmented IP packets received[1]. Generally, packets are not fragmented.
`Malformed IP Packets`	Number of malformed IP packets received[1]. This include runt packets and checksum errors.
`Total Sent Packets`	Total number of packets transmitted from WebGate[1].
`Transmit Collisions`	Not available
`Failed Transmissions`	Not available
`Last TX Carrier`	Not available
`Active FTP Connections`	Number of currently active FTP connections.
`Active HTTP Connections`	Number of currently active HTTP connections.
`Active SNMP Connections`	Number of currently active SNMP connections.

[1]This is a cumulative count that restarts at each WebGate reboot.

Console example:

```
> NetStat↵
Ethernet Statistics:
ARP Packets         : 6
TCP Packets         : 97
UDP Packets         : 1
ICMP Packets        : 0
Other IP Packets    : 0
Other Not IP Packets: 0
IP Fragments        : 0
Malformed IP Packets: 0
Total Sent Packets  : 108
Transmit Collisions : N.A.
Failed Transmissions: N.A.
Last TX Carrier     : N.A.
Active FTP Connections  : 0
Active HTTP Connections : 0
Active SNMP Connections : 0

> _
```

Passwd	•		
Description	*Read*	*Write*	*Saved*
• Enter the user password for the console	×	**Guest**	×

Enter the password to login a user.
The console interface fully comply with the restrictions imposed from the users management system, described in chapter **User**

Management.
To gain access to WebGate through the console interface, a user is required to perform a two steps operation first:

1. Enter his/her name using the ***Login*** command
2. Enter his/her password using the ***Passwd*** command

When a wrong password is entered, an error is issued and the entire process must be repeated from the beginning.
Moreover, please note that the ***login*** command will not issue any immediate error if the entered name is not defined in the users table. The error will be addressed from the next ***passwd*** command.

Syntax:
Passwd = <*Password*>

***Where* <*Password*> represents the user password associated to the previous *Login* Command.**

Returned value:
"User name not defined, you must use <Login> first (code -312)" if another user is already logged in
"Bad username or password (code -313)" if the user name or password are invalid
Welcome string if the password match.

Note:
When WebGate reboots, no user is automatically logged in.
To prevent unauthorized access, users are automatically logged out after 5 minutes from the last console input.
When no user is logged in, the console can be used anyway. However, the access is restricted to the same rights of the "anonymous" user (see "anonymous" User). By default the anonymous user have administrator rights. Consequently, if you don't need any security capability, you will never need to use "login" command.

Console example:

```
> Logout↵

> Login = JohnDoe↵

> Passwd =↵
Welcome JohnDoe! Your access level is 249 (administrator)

> _
```

Reboot	•		
Description	*Read*	*Write*	*Saved*
Restart the WebGate	×	**Administrator**	×

A shutdown procedure is initiated forcibly closing all files and connections, and the WebGate is restarted.
Please note that this operation may take some seconds.

This command is used mainly to perform a firmware upgrade, since the existence of an updated file is checked at each WebGate bootstrap.

Syntax:
Reboot

Console example:

```
> Reboot↵

<… … … bootstrap message … … …>
```

Registers	•		
Description	*Read*	*Write*	*Saved*
Displays a list of all the currently defined registers	✓	×	×

This command lists all the defined arithmetic registers.
Arithmetic registers are explained in chapter **About Expressions, Registers and Functions.**
Please note that some of the listed registers are predefined and not modifiable.

Syntax:
`Registers`

Console example:

```
> set(address)=7↵

> registers↵
       administrator: 200
           developer: 250
               guest: 0
          supervisor: 150
                user: 100
             address: 7

> _
```

Note:
The registers highlighted are reserved. They identify minimum access levels for any type of user and are provided to simplify operations

The registers highlighted in **blue** are reserved. They identify minimum access levels for any type of user.

ROCommunity	•		
Description	*Read*	*Write*	*Saved*
Get/set the Read Only Community Name	✓	**Administrator**	✓

Displays or changes the SNMP *Read Only Community* Name.
Please see chapter **Communities** for a description of SNMP communities.

Syntax (*read*):
`ROCommunity`

Returned value for read:
`<name>`

Where *`<name>`* is the community name

Syntax (*write*):
`ROCommunity = <name>`

Where *`<name>`* is the community name.
The name must be a string composed from alphanumeric characters *only* (0…9, A…Z and a…z). No spacing or punctuation characters are allowed. The maximum name length must not exceed *10* characters.

Default Value:
"public"

Console example:

```
> ROCommunity = controls↵

> ROCommunity↵
controls

> _
```

RWCommunity	•		
Description	*Read*	*Write*	*Saved*
Get/set the Read/Write Community Name	✓	**Administrator**	✓

Displays or changes the SNMP *Read/Write Community* Name.
Please see chapter **Communities** for a description of SNMP communities.

Syntax (*read*):
`RWCommunity`

Returned value for read:
`<name>`

Where *`<name>`* is the community name

Syntax (*write*):
`RWCommunity = <name>`

Where *`<name>`* is the community name.
The name must be a string composed from alphanumeric characters *only* (0…9, A…Z and a…z). No spacing or punctuation characters are allowed. The maximum name length must not exceed *10* characters.

Default Value:
"public"

Console example:

```
> RWCommunity = administ↵

> RWCommunity ↵
administ

> _
```

ScanDevs	•		
Description	*Read*	*Write*	*Saved*
Rescan all the devices connected to the WebGate	×	**Administrator**	×

A complete Carel RS485 network scan is performed.

This command may prove useful to update immediately the information about the connected devices.
For example, when a unit is replace by another one of different type but with the same address (e.g. the unit with address "1" was an IR32, and it is replaced with a MPX).

Please note that since the Carel Network Interface is fully reinitialized, an offline unit will be marked as "never connected".

Syntax:
`ScanDevs`

Returned value:
"Scanning Devices " when operation is in progress
Error string if error occurred

Console example:

```
> ScanDevs ↵
Scanning devices!

> _
```

Set	•		
Description	*Read*	*Write*	*Saved*
Assigns a numeric value to a register	×	**Guest**	×

This function creates an register and assigns the numeric result of an arithmetic expression to it.

Syntax:
Set(*<register>*)=*<expression>*

<register> Register name.
<expression> Arithmetic expression: it may be a number, register, or a combination of the two.
Warning: do not insert blanks inside arithmetic expressions, because they could cause wrong results.

Please see chapter About Expressions, Registers and Functions for an accurate description of registers and arithmetic expressions.

Console example:

```
> Set(Address)=7↵

> Eval(Address – 1)↵
6

> Set(Index)=Address+2↵ ← DO NOT insert blanks here

> Eval(Index – Address + 1)↵
3

> _
```

SetMax	•		
Description	*Read*	*Write*	*Saved*
Set the maximum allowed value for a register	×	**Guest**	×

This function defines a maximum value for a register.

Syntax:
Setmax(*<register>*)=*<expression>*

<register> Register name.
<expression> Arithmetic expression: it may be a number, register, or a combination of the two.
Warning: do not insert blanks inside arithmetic expressions, because they could cause wrong results.

Please see chapter About Expressions, Registers and Functions for an accurate description of registers and arithmetic expressions.

Note:
If a register with the given name was not already defined when "*SetMax*" is called the register itself is created, and the lowest value between "0" and the given maximum is assigned to it.

Console example:

```
> SetMax(Address)=10↵

> Set(Address)=200↵

> Eval(Address)↵
10

> _
```

SetMin	•		
Description	*Read*	*Write*	*Saved*
Set the minimum allowed value for a register	×	**Guest**	×

This function defines a minimum value for a register.

Syntax:
`Setmin(<register>)=<expression>`

<register>	Register name.
<expression>	Arithmetic expression: it may be a number, register, or a combination of the two. **Warning:** do not insert blanks inside arithmetic expressions, because they could cause wrong results.

Please see chapter About Expressions, Registers and Functions for an accurate description of registers and arithmetic expressions.

Note: If a register with the given name was not already defined when "*SetMin*" is called the register itself is created, and the greatest value between "0" and the given minimum is assigned to it.

Console example:

```
> SetMin(Address)=20↵

> Set(Address)=7↵

> Eval(Address)↵
20

> _
```

SNMPSyscont	•		
Description	*Read*	*Write*	*Saved*
Get/set the SNMP System Contact	✓	**Administrator**	✓

Displays or changes the SNMP *System Contact* Name.
Please see chapter **WebGate SNMP Protocol** for a description of SNMP.

This string is used from SNMP Managers.

Syntax (*read*):
`SNMPSyscont`

Returned value for read:
`<name>`

Where *`<name>`* is the system name

Syntax (*write*):
`SNMPSyscont = <name>`

Where *<name>* is the system contact name.
The name must be a string composed from alphanumeric characters *only* (0…9, A…Z and a…z). No spacing or punctuation characters are allowed.
The maximum name length must not exceed *80* characters.

Default Value:
"*`Undefined`*"

Console example:

```
> SNMPSyscont = JohnSmith↵

> SNMPSyscont↵
JohnSmith

> _
```

SNMPSysloc	•		
Description	*Read*	*Write*	*Saved*
Get/set the SNMP System Location	✓	**Administrator**	✓

Displays or changes the SNMP *System Location* string.
Please see chapter **SNMP** for a description of SNMP.

This string is used from SNMP Managers.

Syntax (*read*):
`SNMPSysloc`

Returned value for read: ***`<location>`***

Where *`<location>`* is the system location

Syntax (*write*):
`SNMPSysloc = <location>`

Where *`<location>`* is the system location.
The location must be a string composed from alphanumeric characters *only* (0…9, A…Z and a…z). No spacing or punctuation characters are allowed. The maximum string length must not exceed *80* characters.

Default Value:*"Undefined"*

Console example:

```
> SNMPSysloc = Roof↵

> SNMPSysloc ↵
Roof

> _
```

SNMPSysname	•		
Description	*Read*	*Write*	*Saved*
Get/set the SNMP System Name	✓	**Administrator**	✓

Displays or changes the SNMP *System Name* string.
Please see chapter **WebGate SNMP Protocol** for a description of SNMP.

This string is used from SNMP Managers.

Syntax (*read*):
`SNMPSysloc`

Returned value for read: *`<name>`*

Where *`<name>`* is the system location

Syntax (*write*):
`SNMPSysloc = <name>`

Where *`<name>`* is the system name.
The name must be a string composed from alphanumeric characters *only* (0…9, A…Z and a…z). No spacing or punctuation characters are allowed. The maximum string length must not exceed *40* characters.

Default Value:
"CarelWebgate"

Console example:

```
> SNMPSysname = PlantChillersSystem↵

> SNMPSysname ↵
PlantChillersSystem

> _
```

SNMPAgentRel	•		
Description	*Read*	*Write*	*Saved*
Return the Agent software release	✓	×	×

This function returns the SNMP Agent software version.

Syntax:
`SWVersion`

Returned value:
`<Version>`
`<Version>` **is used to identify a major software change**

Console example:

```
> SNMPAgentRel↵
10

> _
```

Status	•		
Description	*Read*	*Write*	*Saved*
Displays the current WebGate status	✓	×	×

This command return a brief WebGate status report.

The use of this command is intended mainly for Carel internal debug purposes.

Syntax:
`Status`

Console example:

```
> Status↵
WebGate Status Summary:
Access Level: 250
root code begins at 00:0000, ends at 00:6993
root data begins at 92:cbff, ends at 92:7d32
xmem code begins at f9:e000, ends at 2c:eec4
stack begins at d000, ends at dfff
largest xalloc block available: 9656 bytes
LWM configuration file valid.
number of open files: 0
free timers: 10

> _
```

Status485	•		
Description	*Read*	*Write*	*Saved*
Get information about the RS485 and Carel Network interface	✓	×	×

This command return a syntetic RS485 interface status line. It can be useful when a RS485 configuration trouble is suspected.

Syntax:
`Status485`

Returned value:
The explanatory string will be one of the following:

- "*RS485 settings are valid. Baudrate selected: xxx*" when no error occurred, or
- "*Error during LWM Table compilation!*" if the low-level devices configuration file (LWM) is not valid. If this is the case, contact Carel for to obtain an updated configuration file.
- An indication of wrong baudrate selection

Console example:

```
> Status485↵
RS485 settings are valid.
Baudrate selected: 19200

> _
```

SWVersion	•		
Description	*Read*	*Write*	*Saved*
Return the WebGate software version	✓	×	×

This function returns the latest WebGate software update version (revision).
Please note that only *firmware* updates are tracked. File System updates don't change this number.

Syntax:
`SWVersion`

Returned value:
`<Version>`*`.`*`<Release>`* **(build** *`<build>`), created on** *`<date>`*

`<Version>`	**is used to identify a major software change**
`<Release>`	**is used to identify a minor software change**
`<Build>`	**indicates the software build.**
`<Date>`	**the complete date *when* the software was put together**

Console example:

```
> SWVersion↵
0.8 (build 28), created on 18/06/2002 11:15:07

> _
```

TrapCommunity	•		
Description	*Read*	*Write*	*Saved*
Get/set the Trap Community Name	✓	**Administrator**	✓

Displays or changes the SNMP Community Name for traps sent.
Please see chapter **Communities** for a description of SNMP communities.

Syntax (*read*):
`TrapCommunity`

Returned value for read:
`<name>`

Where *`<name>`* is the community name
Syntax (*write*):
`TrapCommunity =` ***`<name>`***

Where *<name>* is the community name.
The name must be a string composed from alphanumeric characters *only* (0…9, A…Z and a…z). No spacing or punctuation characters are allowed. The maximum name length must not exceed *10* characters.

Default Value:
"public"

Console example:

```
> TrapCommunity = trapcom↵

> TrapCommunity↵
trapcom

> _
```

TrapIPAddress	•		
Description	*Read*	*Write*	*Saved*
Get/set the trap manager IP address	✓	**Administrator**	✓

Displays or changes the SNMP trap manager IP address.
Please see chapter **Communities** for a description of SNMP communities.

Syntax (*read*):
`TrapIPAddress`

Returned value for read:
<IPx>.<IPy>.<IPz>.<IPw>

Where *<IPx>*, *<IPy>*, *<IPz>* and *<IPw>* are the four bytes of IP expressed using the standard dotted format string.

Syntax (*write*):
`TrapIPAddress = <IPx>.<IPy>.<IPz>.<IPw>`

Where `<IPx>`, `<IPy>`, `<IPz>` and `<IPw>` are the four bytes of IP expressed using the standard dotted decimal format string.

Allowed values: Any combination of four numbers in the range 0…255.
To disable trap transmissions from WebGate, set this value to 255.255.255.255

Note: When you change Trap IP address, WebGate will immediately store the value just selected. However, the old value will be used until a SNMP connection is still active.

Returned value for write:
One of the following situations may occur:
Valid value entered and no connections still active: `IP value updated!`
Valid value entered and one or more connections still active: `Valid for next connection!`
Invalid value entered: "bad value" or "syntax error" string.

Default Value:
255.255.255.255 (disabled)

Console example:

```
> TrapIPAddress=192.168.0.25↵
IP value updated!

> TrapIPAddress↵
192.168.0.25

> _
```

UpdateStatus	•		
Description	*Read*	*Write*	*Saved*
Return the last update error code	✓	×	×

This command return a string specifying the error occurred (if any) when the last reboot was performed and a firmware update was tried.

Please note that the existence of an updated file is checked at every WebGate reboot, and *not* during normal operation. The update procedure is described in chapter **Firmware Upgrade**.

Syntax:
`UpdateStatus`

Returned value:

- "*Update file not found*" if no file was found. ***This is the normal behaviour and not an error***, but simply an advise that no update file was found. Furthermore, ***please note that after any successful upgrade the used update file is deleted to free up some space for new user files***.
- "*Not a valid WebGate update file*" or "*Incompatible update file*" when a file seems devised to upgrade but prove itself not adequate for WebGate. You may have uploaded an update file for a different device other than WebGate.
- "*Update file corrupted*" when the file is damaged. Contact Carel to obtain a functional file.

Console example:

```
> UpdateStatus↵
Update file not found (code -900)

> _
```

UserLevel	•		
Description	*Read*	*Write*	*Saved*
Get/set the given access level	✓	**Guest**	✓

Displays or changes the access level of the given user.
Please see chapter **User Management** for a description of user management system.

Syntax (*read*):
`UserLevel(<index>)`

Where `<index>` represents the user index in the users table (0…5)

Returned value for read: `<level>`

Where `<level>` is a number in the range 0…249 indicating the access level of the given user.

Syntax (*write*):
`UserLevel(<index>) = <level>`

Where `<index>` represents the user index in the users table (0…5), and
`<level>` is a number in the range 0…249 indicating the access level of the given user.

Please note that operators are only allowed to change information of users with their same or lower access level. For example, an *administrator* (level 200…249) can change *supervisors* information (level 150…199) , but a *supervisor* cannot change the access level of any *administrator*.
Furthermore, the access level set for an user cannot be greater than the current access level of the operator, stated from the "`accesslevel`" command. This is necessary to avoid that an operator arbitrarily elevate his proper access level.

Suggested Values:
To simplify users management, please use only the following suggested values:

Desired Level	Suggested Value
guest	**99**
user	149
supervisor	**199**
administrator	249

Default Value:
***"administrator"* (249)**

Console example:

```
> UserLevel(3) = 99↵

> UserLevel(3)↵
99

> _
```

UserLevelString	•		
Description	*Read*	*Write*	*Saved*
Return the minimum access level descriptive string for a given user	✓	×	×

Displays a string describing the access level of the given user.
Please see chapter **User Management** for a description of user management system.

Syntax:
UserLevelString(*<index>*)

Where *<index>* represents the user index in the users table (0…5)

Returned value for read:
<levelstring>

Where *<levelstring>* is a string indicating the access level of the given user as indicated in the table below:

User Level	Descriptive String
From 0 to 99	**guest**
From 100 to 149	user
From 150 to 199	**supervisor**
From 200 to 249	administrator

Console example:

```
> UserLevel(3) = 99↵

> UserLevelString(3)↵
guest

> _
```

UserName	•		
Description	*Read*	*Write*	*Saved*
Get/set the given user name	✓	**Guest**	✓

Displays or changes an user name.
Please see chapter **User Management** for a description of user management system.

Syntax (*read*):
UserName(*<index>*)

Where *<index>* represents the user index in the users table (0…5)

Returned value for read:
<name>

Where *<name>* is the user name.

Syntax (*write*):
UserName(*<index>*) = *<name>*

Where *<index>* represents the user index in the users table (0…5), and
<name> is the new user name.

The name must be a string composed from alphanumeric characters *only* (0…9, A…Z and a…z). No spacing or punctuation characters are allowed. The maximum name length must not exceed *20* characters.

If the given name is "*anonymous*", the access rules of that table entry will be applied to any anonymous access (from HTTP, FTP or console).

If you want to erase an user and left the table entry empty, simply use an empty string for the name (see the console example below).

Please note that operators are only allowed to change information of users with their same or lower access level. For example, an *administrator* (level 200…249) can change *supervisors* information (level 150…199), but a *supervisor* cannot change the user name of any *administrator*.

Also note that when more than a table entry refers to the same user name, the one with the lowest index will be used. For example, if "*UserName(2)=John*" and "*UserName(4)=John*", access level and password will be bring from "*UserLevel(2)*" and "*UserPwd(2)*".

Default Value:
***Empty string* (unused)**

Console example:

```
> UserName(3) = John↵

> UserName(3)↵
John

> UserName(4) =↵ ← Empty string: the entry nº 4 is freed

> UserName(4)↵


> _
```

UserPwd	•		
Description	*Read*	*Write*	*Saved*
Get/set the given user password	✓	**Guest**	✓

Displays or changes an user password.
Please see chapter **User Management** for a description of user management system.

Syntax (*read*):
UserPwd(*<index>*)

Where *<index>* represents the user index in the users table (0…5)

Returned value for read:
<password>

Where *<password>* is the user password.

Please note that operators are only allowed to view passwords of users with their same or lower access level. If this is not the case, a line of asterisks will be shown.

Syntax (*write*):
UserPwd(*<index>*) = *<password>*

Where *<index>* represents the user index in the users table (0…5), and
<password> is the new user name.

The password must be a string composed from alphanumeric characters *only* (0…9, A…Z and a…z). No spacing or punctuation characters are allowed.
The maximum password length must not exceed *20* characters.

If the given password is an empty string, the user will access to WebGate with any string in place of the password (is pointed out that when using the console, the "*passwd=*" command must be used anyway).

Please note that operators are only allowed to change information of users with their same or lower access level. For example, an *administrator* (level 200…249) can change *supervisors* information (level 150…199), but a *supervisor* cannot change the password of any *administrator*.

Default Value:
***Empty string* (no password required)**

Console example:

```
> UserPwd(3) = Hello↵

> UserPwd(3)↵
Hello

> UserPwd(4) =↵ ← Empty string: the user nº 4 will enter with any password

> _
```

Var	•		
Description	*Read*	*Write*	*Saved*
Get/set a device variable value	✓	**Supervisor**	×

This function displays or changes a variable of a Carel unit connected to the RS485 interface.

Syntax (*read*):
`Var(<address>, <type>, <index>)`

Where *<address>* represents the unit address (from "1" up to the value indicated from the "`MaxDevs`" function),
<type> is the type of variable to read. It can be one of the following values:
"1": digital
"2": analog
"3": integer
<index> index of the variable in the unit

Returned value for read:
One of the following values is returned:
- value of the required variable if ready and no error occurred
- warning message if the value is not available, for example:
"*Please wait, checking status!*" or "*variable not yet updated.*" if the variable was not acquired yet,
"*Unit OFF-LINE: unreliable value!*" if the device is not online
- an error message if a bad parameter value is entered

Syntax (*write*):
`Var(<address>, <type>, <index>, <min>, <max>) = <value>`

Where *<address>* represents the unit address (from "1" up to the value indicated from the "MaxDevs" function),
<type> is the type of variable to write. It can be one of the following values:
"1": digital
"2": analog
"3": integer
<index> index of the variable in the unit
<min> minimum value allowed. This is required, when using HTML pages, to avoid device misconfiguration of the device. The minimum value allowed is indicated in the user manual of the specific device. Don't use a lower value, since this may cause unit malfunction.
<max> maximum value allowed. This is required, when using HTML pages, to avoid device misconfiguration of the device due to involuntary user errors. The maximum value allowed is indicated in the user manual of the specific device. Don't use a higher value, since this may cause unit malfunction.
<value> value to assign to the variable

Returned value for write:

One of the following values is returned:
"*Write operation terminated!*" if the value was immediately updated or was the same value read before.
"*Message sent. Read new value*" if the value is accepted and is going to be sent to the unit. Since the transmission is performed in background to avoid system slow-downs, using console the user should read the value after a while to make sure the operation was completed. When using HTTP, generally the page is not entirely transmitted until the operation is completed (see *Using Text-Boxes and Buttons inside forms*).
"*Variable read only: write operation aborted!*" se la variabile non è accessibile in scrittura if the variable cannot be accessed in writing. (This may be the case, for example, of a probe readout).
Another error message appears if a bad parameter or bad assignment value is entered.

Console example:

```
> Var(1,2,3)↵
33.0

> Var(1,2,3,0,50)=20↵
 Message sent. Read new value

> _
```

WakeOn	•		
Description	*Read*	*Write*	*Saved*
Displays the time elapsed since the last WebGate reboot	✓	✖	✖

Return a string indicating the time elapsed since the last WebGate reset.

Syntax:
WakeOn

Returned value:
Time elapsed from last reset: <days> day, <hr> hour, <min> min, <sec> sec

Console example:

```
> WakeOn↵
Time elapsed from last reset: 0 day, 2 hour, 51 min, 33 sec

> _
```

12. About Expressions, Registers and Functions

To provide HTML pages with greater programming flexibility, some functions allows the use of simple integer arithmetic expression in their parameters or assignments.
Arithmetic expressions can consist of numbers, registers (explained in the paragraph below), or a combination of the two in the following manner:

Number	- example: **`Var(1, 1, 1)`**
Register	- example: **`Var(1, 1, Index)`**
Complex expression	- example: **`Var(Address - 1, Type, Index + 1)`**

Graphically, an expression could be indicated as below:

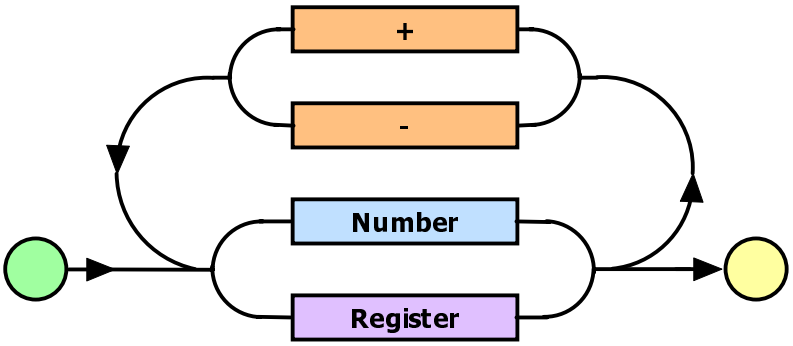


Fig. 12.1

The principal functions that allow the use of expressions in their parameters are:

- **Var** (for `<address>`, `<type>` and `<index>` parameters)
- **Eval**
- **UserLevel**
- **UserLevelString**
- **UserName**
- **UserPwd**

The principal functions that allow the use of expressions for the assignment are:

- **Set**
- **SetMin**
- **SetMax**
- **UserLevel**

12.1 Registers

A "register" is an alphabetic, non case-sensitive string of at most 16 characters, representing a signed integer number comprised between $-2^{-31}+1$ and $2^{31}-1$.
To create a register use the "**Set**", "**SetMin**" or "**SetMax**" functions.

When using registers, please consider the following points:

- Registers are never allowed to assume values exceeding the limits imposed by "*SetMin*" and "*SetMax*" functions. If an expression results in a value lower or greater it will be limited.
- In addition to the predefined read only registers, **at most 8 user defined registers are allowed at the same time**. When the *9th* register is created, the oldest one is discarded.
- **Registers are a shared and unprotected resource.** No access level is required to read or write to any of these. They can be accessed and discarded at any time from any HTML page or from the console.
  Moreover, **care must be taken because two or more users accessing to pages using the same registers may interfere each other.** This may occur only in rarely occasions but since inevitable, make sure to avoid such situations (for example, use registers only when absolutely necessary and only when a single user have the password to access to those pages).
- Registers are reset after a WebGate reboot.

12.2 Some additional notes about arithmetic expressions:

- Only the sum "+" and subtraction "-" operators are allowed.
- Numeric values exceeding the range **$-2^{31}+1 \ldots 2^{31}-1$** resulting from user input or calculation are not evaluated correctly.
- The use of an undefined register name results in a syntax error.

13. Technical Specifications

Power Supply: 18Vac -15%…+10%, 50/60Hz or 24Vac -15%…+10%, 50/60Hz
Rated Power: 3W

• **For desktop installation:** from mini AC power jack, dia. 5mm. Use ONLY the power adapter supplied on request by Carel P.N. TRA1806ITA. The use of different power adapters may damage the hardware.

• **For panel mounting:** from 2-pole removable horizontal terminal blocks (spacing 5.08mm), lead size 0.5-1.5mm2. Characteristics of the fuse, obligatory, to be inserted between the WebGate power supply input and the power supply transformer: 500mAT.
Use a class 2 dedicated safety transformer rated to at least 6VA. Warning: do not earth the secondary.
The use of the power adapter TRA1810DIN supplied on request by Carel is suggested.

Interfaces:
• Serial RS485 opto-insulated, 3-pole removable horizontal terminal blocks (spacing 3.81mm), lead size: 0.14-1.5mm2. Use a twisted pair shielded cable AWG20-22, max 1000m, capacitance between the cables <90 pF/m.
• Serial RS232 DTE interface, 9-pin male DB-9 connector; 19200 baud (configurable), 8 data bits, 1 stop bit, no parity and no flow control. Use a null-modem shielded cable, max. 10m, cable capacity <2500pF.
• Ethernet interface, RJ-45 connector for 10BaseT Ethernet.
Use a class 5 shielded cable, max. 100m.

Functional Characteristics:
Standard Internetworking Protocols: SNMP v1, HTTP, FTP
Memory: 128KB RAM, 1MB Flash (400KB available for web pages and user data).
File System: 100 files max. available to the user. Minimum size used from a file: 1KB

Other:
Operating Conditions: 0T50°C, 0 up to 90% rH non condensing
Storage Conditions: -10T70°C, 0 up to 90% rH non condensing
Index of Protection (IP): IP20
Dimensions (mm): 152 x 128 x 39
Environmental pollution: normal
Category of resistance to fire and heat: D
Software class and structure: A

Disposal of the Product:
The product is made of electronic, metal and plastic parts. Such parts must be disposed of in accordance with the laws in force in each country.

CAREL

Technology & Evolution

CAREL S.p.A.
Via dell'Industria, 11 - 35020 Brugine - Padova (Italy)
Tel. (+39) 049.9716611 Fax (+39) 049.9716600
http://www.carel.com - e-mail: carel@carel.com

Agency*:*

Cod: +030220230
preliminary version dated 28 november 2002