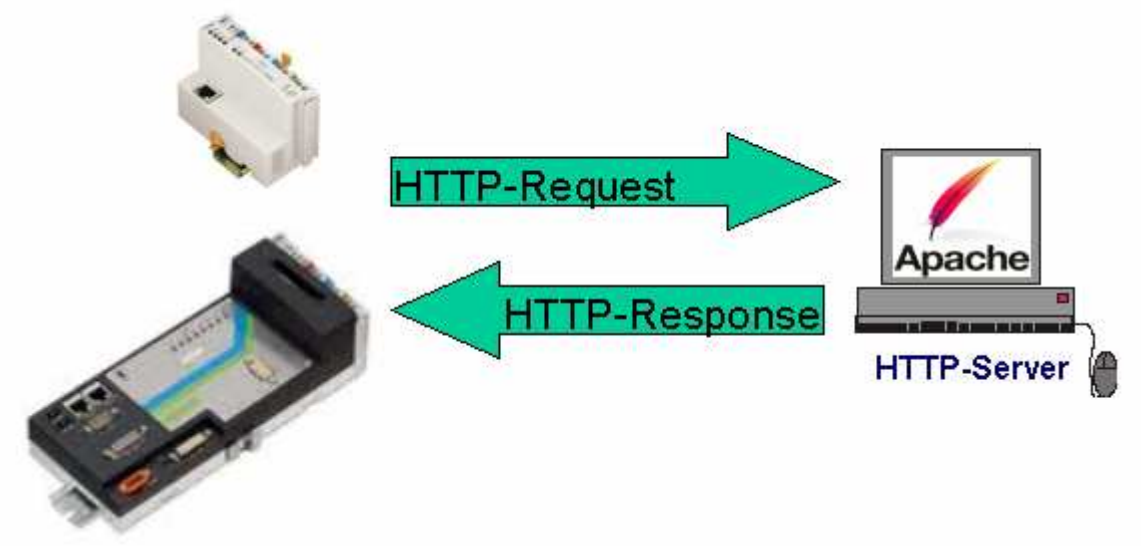
WAGO I/O SYSTEM

Using library ”WagoLibHttp_02.lib” to communicate with remote web server



Application note

A303000, English
Version 1.0.0

WAGO®
INNOVATIVE CONNECTIONS

WAGO Kontakttechnik GmbH & Co. KG
Hansastraße 27
D-32423 Minden

Phone: +49 (0) 571/8 87 – 0
Fax: +49 (0) 571/8 87 – 1 69

E-Mail: info@wago.com

Web: http://www.wago.com

Technical Support
Phone: +49 (0) 571/8 87 – 5 55
Fax: +49 (0) 571/8 87 – 4 30

E-Mail: support@wago.com

Every conceivable measure has been taken to ensure the correctness and completeness of this documentation. However, as errors can never be fully excluded we would appreciate any information or ideas at any time.

We wish to point out that the software and hardware terms as well as the trademarks of companies used and/or mentioned in the present manual are generally trademark or patent protected.

TABLE OF CONTENTS

1 Important comments

To ensure fast installation and start-up of the units described in this manual, we strongly recommend that the following information and explanation is carefully read and adhered to.

1.1 Legal principles

1.1.1 Copyright

This manual is copyrighted, together with all figures and illustrations contained therein. Any use of this manual which infringes the copyright provisions stipulated herein, is not permitted. Reproduction, translation and electronic and photo-technical archiving and amendments require the written consent of WAGO Kontakttechnik GmbH & Co. KG. Non-observance will entail the right of claims for damages.

1.1.2 Personnel qualification

The use of the product detailed in this manual is exclusively geared to specialists having qualifications in PLC programming, electrical specialists or persons instructed by electrical specialists who are also familiar with the valid standards. WAGO Kontakttechnik GmbH & Co. KG declines all liability resulting from improper action and damage to WAGO products and third party products due to non-observance of the information contained in this manual.

1.1.3 Intended use

For each individual application, the components supplied are to work with a dedicated hardware and software configuration. Modifications are only admitted within the framework of the possibilities documented in the manuals. All other changes to the hardware and/or software and the non-conforming use of the components entail the exclusion of liability on part of WAGO Kontakttechnik GmbH & Co. KG.

Please direct any requirements pertaining to a modified and/or new hardware or software configuration directly to WAGO Kontakttechnik GmbH & Co.KG.

1.2 Range of validity

This application note is based on the stated hardware and software of the specific manufacturer as well as the correspondent documentation. This application note is therefore only valid for the described installation.

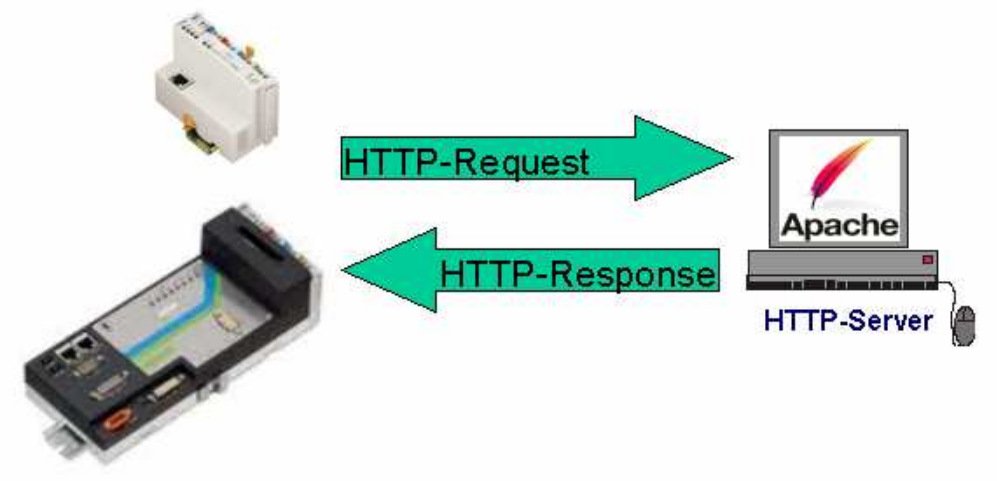
New hardware and software versions may need to be handled differently. Please note the detailed description in the specific manuals.

1.3 Symbols

	**Danger** Always observe this information to protect persons from injury.
	**Warning** Always observe this information to prevent damage to the device.
	**Attention** Marginal conditions must always be observed to ensure smooth operation.
	**ESD** (Electrostatic Discharge) Warning of damage to the components by electrostatic discharge. Observe the precautionary measure for handling components at risk.
	**Note** Routines or advice for efficient use of the device and software optimisation.
	**More information** References to additional literature, manuals, data sheets and INTERNET pages

2 Introduction

This application note describes use of Hypertext Transfer Protocol (HTTP) library “WagoLibHttp_02.lib”. The focus of this note points to the transfer of request data from a process automation device i.e. WAGO Ethernet-Controller to an HTTP-Server via the HTTP protocol and further how to process the server response directly in the controller.



HTTP is a networking protocol for information systems and defines a request-response protocol in the client-server computing model. HTTP specifies several request methods indicating the desired action to be performed on a resource identified by Uniform Resource Locators (URLs). This document shows GET and POST methods use.

- HTTP-GET requests a representation of the specified URL, if data is sent then as a part of the URL. Data length is restricted.
- HTTP-POST is used to submit data to be processed to an URL. The data is included in the body of the request. Maximum length of data is restricted only by TCP-Sockets transmit buffer size (i.e. 60kB for the 750-841 Wago coupler).

Presented examples are as follows

- **Example_01: Read HTML or XML file** shows how to load and parse a HTML page and how to use the output in the PLC code.
- **Example_02: Weather forecast HTTP-API service** presents the World Weather Forecast service data negotiation
- **Example_03: Data logging to WAGO-TO-PASS portal** shows how to log a process data to the WAGO-TO-PASS portal
- **Example_04: WAGO-TO-PASS multiple request** shows how to store wide input data set to WAGO-TO-PASS using multiple HTTP POST request

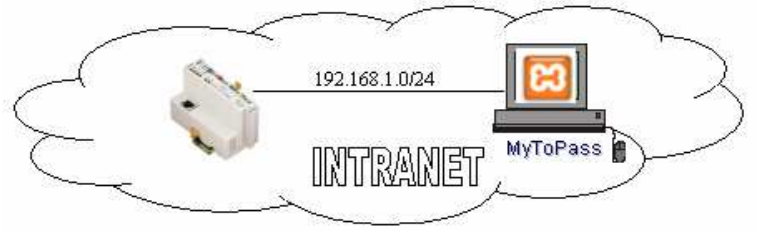
- **Example_05: Own HTTP-API using PHP** presents how to create your own customer specific web API using PHP scripting language
- **Example_06: Database access via HTTP-API** shows how to process the interaction between a PLC program and a database engine MySQL
- **Example_07: Dynamic DNS service DynDNS** presents the DynDNS service (http://www.dyndns.com) in use.

2.1 Network configuration

Presented examples use both the Local Area Network (LAN) and the Wide Area Network (WAN) configuration. Possible settings are presented in this chapter.

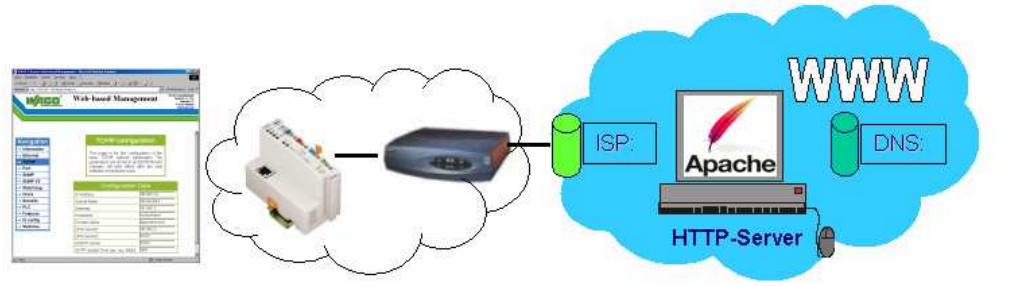
Local Area Network (LAN)

Easiest configuration is Ethernet-Controller and Web-Server operates in the same LAN, known as the Intranet. In this scenario no additional configuration is needed.



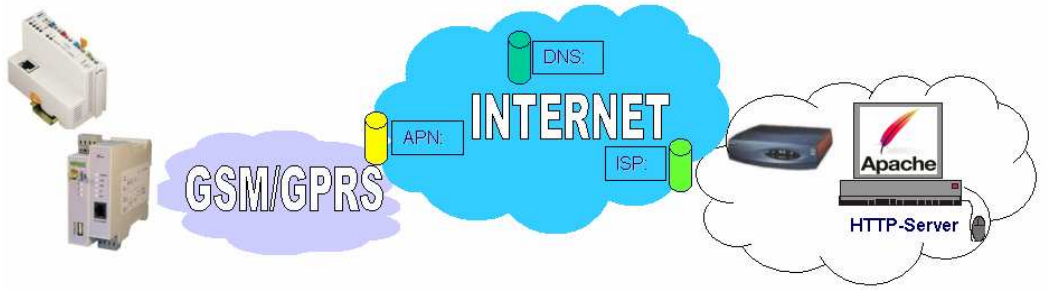
Wide Area Network (WAN)

When HTTP-Server operates in World Wide Web (WWW) also known as WAN a router is needed to connect the local network via a Internet-Service-Provider(ISP) with the internet.



To inform the Ethernet-Controller about the router, apply routers IP-Address as “Gateway” and “DNS-Server” in Web-Based-Management (WBM). Together with WAGO-TO-PASS-Router 761-520 you can set up a Wireless WAN-Solution.

It is also possible to host the web server behind your local router.



Accessibility from the internet requires additional configuration, such as “Port forwarding” from your NAT-Router to the XAMPP server, and DynDNS account like “myWebServer.dyndns.org” which makes your router accessible from the internet when your internet provider only assigns dynamic IP addresses.

In each of these cases, the basic functionality is the same. A client sends a HTTP request (together with process data) to a specialized script on a web server. The web server processes the request and answers with a HTTP response.

Troubleshooting

Possible problems, like non reachable Internet access, silent DNS server response etc., can be caused by wrong setting of the network parameters in each participated device. Some hints to solve the problems are described in “Annex B: Network problem analysis” chapter.

2.2 Test environment

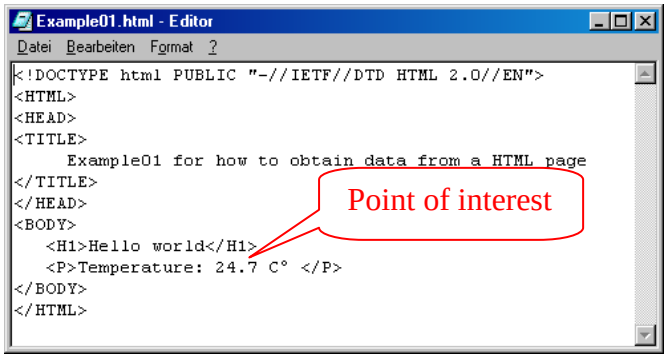
Almost all examples need an access to the WAN, some of them requires running an HTTP-Server, PHP engine and MySQL database environment on the server side.

For the testing purposes install XAMPP. The XAMPP is a free and open source cross-platform web server package. It consists mainly of the Apache HTTP Server, MySQL database, interpreters for scripts written in the PHP scripting language etc. The installation process is described either at http://www.apachefriends.org/ or in “Annex A: Test web server environment setup” chapter.

Example_01: Read HTML or XML file

This example shows a basic functionality of “WagoLibHttp_02.lib”.

Let us suppose a standard HTML page which contains information e.g. about temperature. The example aim is to obtain the HTML page and parse the temperature information from the source code. The result is further used for a processing inside the PLC program.

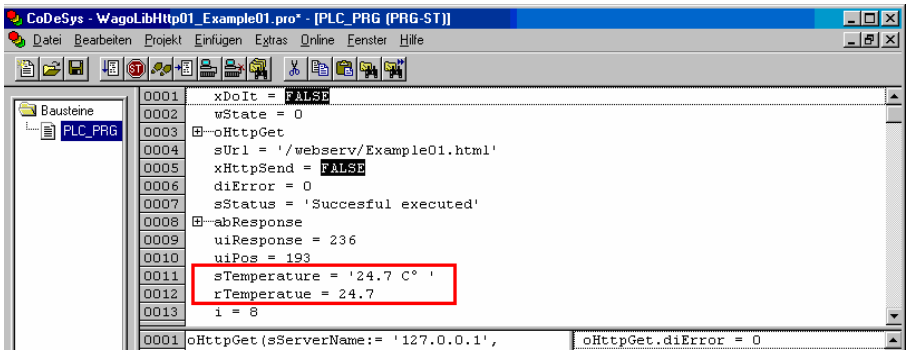


The example **WagoLibHttp02_Example01.pro** communicates with a local web server, which is a fixed part of WAGO Ethernet-Controllers firmware. The example will work with any other web server in LAN or WAN but it requires either installation of the elaborated environment or setup of an access to WAN (router setup etc.).

To get example program working, use an FTP client and copy or create the file "/webserv/Example01.html" in the controller. Note the Wago 750-841 requires the "Basic-Access-Authentication" to access the folder ”/webserv/*”. Check the credentials in the program.

```
oHttpGet(sServerName:= '127.0.0.1',
         wServerPort:= 80,
         sUserName:= 'user',
         sPassword:= 'user',
         pabUrlData:= ADR(sUrl),
         uiUrlLength:= LEN(sUrl),
```

To start the program use “xDoIt” variable, the program parses the temperature string from the page HTML resource code and stores the value to “rTemperature” variable.



This solution can be used for a general remote XML file parsing as well.

Example_02: Weather forecast HTTP-API service

Many web providers offer their HTTP-API to wide public either to process a service or to disseminate important information. The weather forecast is one of the most popular information presented via such interface, for example the World Weather Online service (www.worldweatheronline.com) provides it free of charge.

Since the data is usually processed automatically the device readable form of data is expected (XML, RSS, SOAP, JSON etc.). One of the World Weather Online format is the XML.

WagoLibHttp02_Example02.pro presents the World Weather Forecast service data negotiation, simple parsing of the XML forecast response and propagation of the result to the program code.

Please note the example requires working Internet access. Take care of the controller network settings having valid “Gateway” and “DNS-Server” parameters.

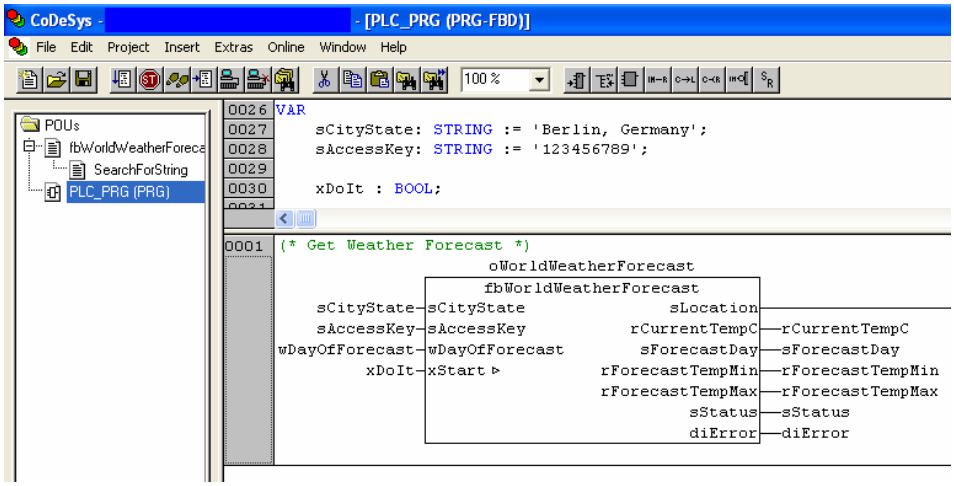
The World Weather Forecast service API HTTP-GET request is as follows

```
http://free.worldweatheronline.com/feed/weather.ashx?q=<city_state>
&format=xml&num_of_days=5&key=<api_key>
```

Where `<city_state>` defines city name and state text code, e.g *berlin,ger* or *paris,france* . `<api_key>` is a user API key obtained via The World Weather registration, use of empty or false key cause no response from the server.

Server response is a XML formatted stream. The XML file example is attached to the example folder project, we use just a subset of XML schema marked in bold. The tags of interest are as follows

```
<?xml version="1.0" encoding="UTF-8"?>
<data>
    <request>
        ...
        <query>Berlin, Germany</query>
    </request>
    <current_condition>
        ...
        <temp_C>8</temp_C>
        ...
    </current_condition>
    ...
    <weather>
        <date>2011-02-08</date>
        <tempMaxC>6</tempMaxC>
        ...
        <tempMinC>1</tempMinC>
        ...
    </weather>
    ...
</data>
```



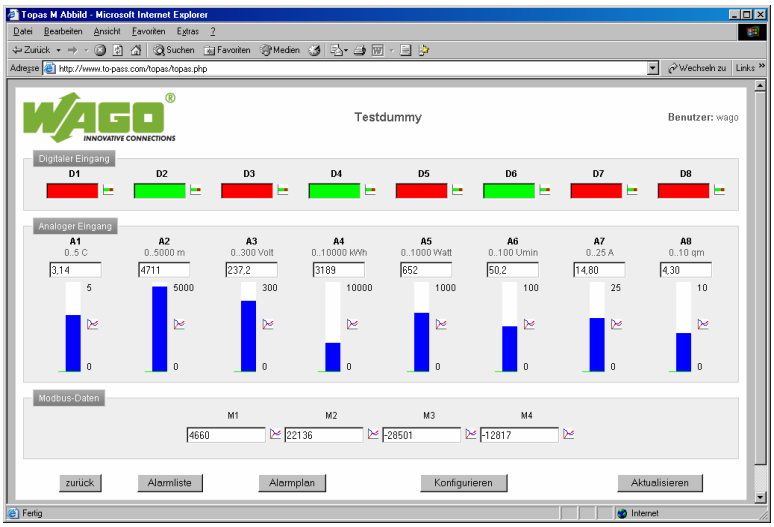
To start the example, set city of interest in “sCityState” and API key in “sAccessKey”, to execute it use “xDoIt” variable. To see forecast for further days set the “wDayOfForecast” to higher number – the API offers 4 day forecast.

For the service documentation see http://www.worldweatheronline.com/feed-generater.aspx web page.

Other weather forecast API providers are for example the YahooWeather (http://www.yahoo.com) or Weather Under Ground (http://www.wunderground.com/) services.

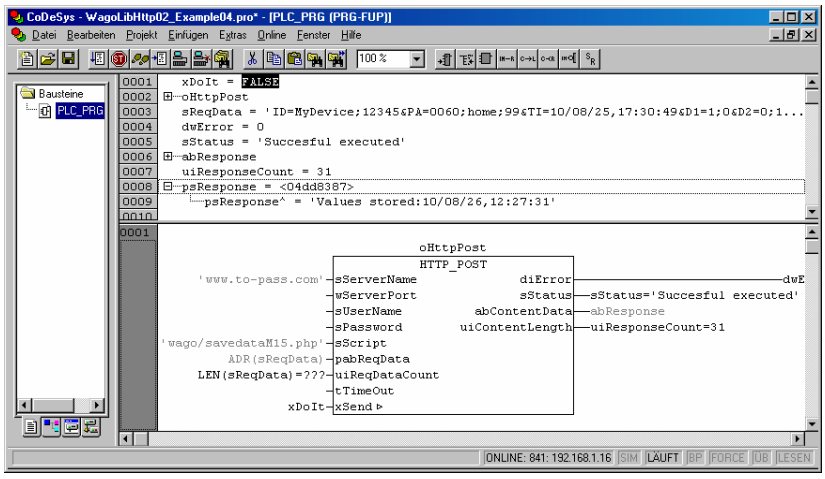
Example_03: Data logging to WAGO-TO-PASS portal

This example shows how to log a process data to the WAGO-TO-PASS portal API. The portal, located at http://www.to-pass.com, is a web based service providing a user interface for viewing and analyzing data logged from remote devices. It offers an API interface to log data via HTTP POST request response method.



Please note the example requires well set internet access, take care of controller network settings e.g. “Gateway” and “DNS-Server”.

The example program uses function block HTTP_POST to send actual process data to WAGO-TO-PASS portal via an API gate provided by PHP script “wago/savedata.php”.



The PLC process data are submitted as parameters of a HTTP POST request packet to WAGO-TO-PASS portal. Portal HTTP server processes the request via the API PHP script using PHP-Engine. Further, the PHP engine forwards received parameters to the portal database and prepare a response to sent back to the PLC.

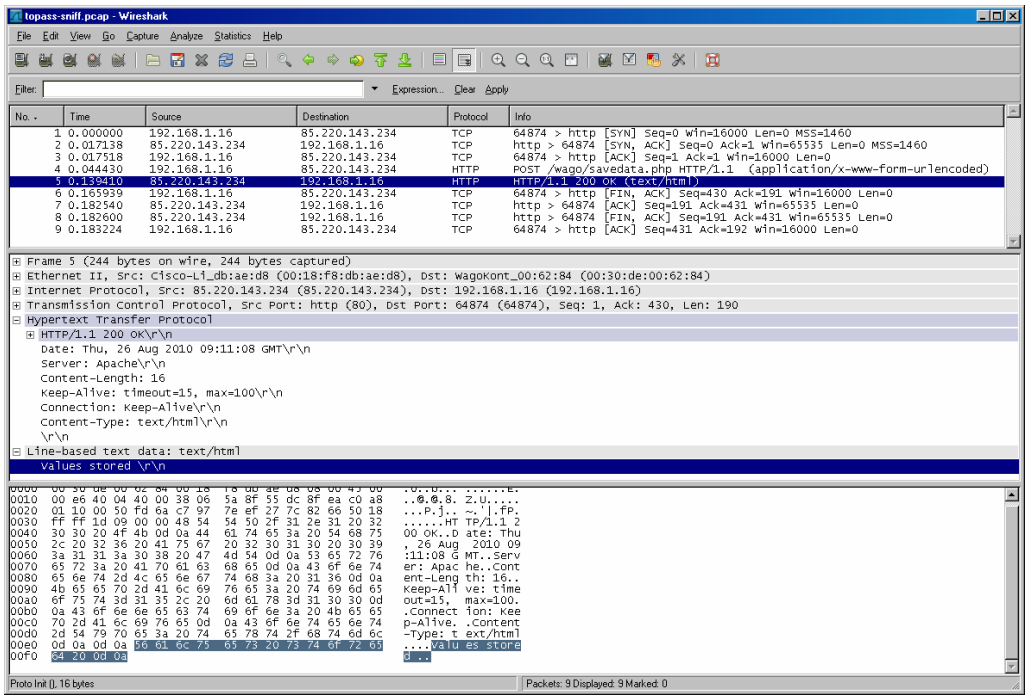
PLC sends following HTTP-POST-Request to WAGO-TO-PASS portal:

```
POST /wago/savedata.php HTTP/1.1
Host: www.to-pass.com
User-Agent: WAGO 750-841
Connection: Keep-Alive
Content-TYPE: application/x-www-form-urlencoded
Content-length: 251

ID=MyDevice;12345&PA=0060;home;99&TI=10/08/25,17:30:49
&D1=1;0&D2=0;1&D3=1;0&D4=0;0&D5=1;1&D6=0;0&D7=1;0&D8=1;1
&A1=3.14;.C;0&A2=4711;%;1&A3=237.2;Volt;0&A4=3189;kWh;0
&A5=652.1;Watt;1&A6=50.2;Umin;0&A7=14.8;Amper;0&A8=4.3;qm/h;1
&MV=1234;5678;90AB;CDEF;AFFE
```

The API PHP script “wago/savedata.php” parses the data parameters and responds with HTTP-POST-Response including “Values stored” string in the case of success to PLC:

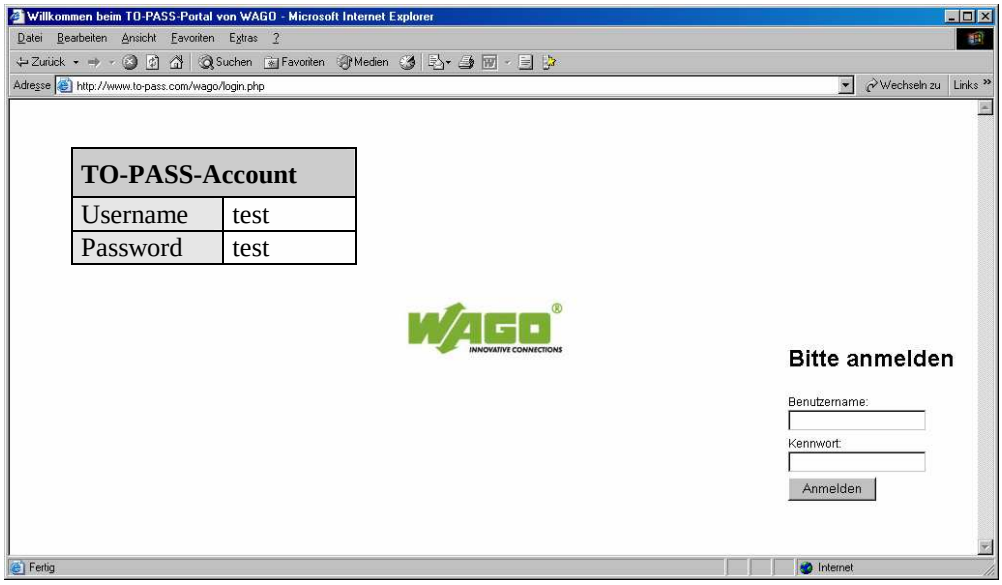
```
HTTP/1.1 200 OK
Date: Thu, 26 Aug 2010 08:55:13 GMT
Server: Apache
Content-Length: 16
Keep-Alive: timeout=15, max=100
Connection: Keep-Alive
Content-Type: text/html

Values stored
```

To analyse the communication between PLC and portal more deeply use the “Wireshark” tool (http://www.wireshark.org). File “topass-sniff.pcab” attached to this application note example presents packet record of success communication.

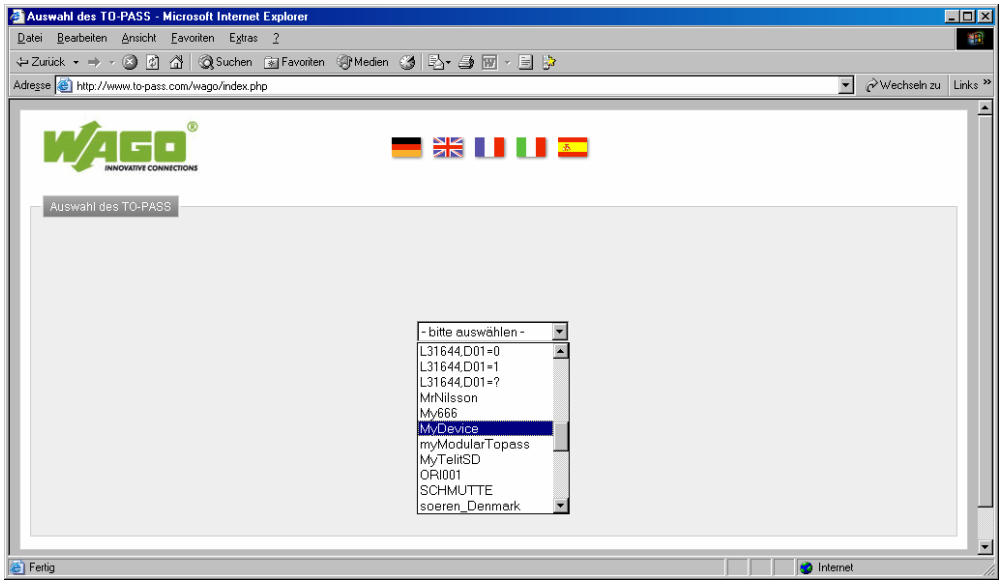


WAGO TO-PASS portal

To take a look onto stored data visit the WAGO-TO-PASS portal (www.to-pass.com). Log in, i.e. click on the “My TO-PASS” button and login as “test” user with “test” password.

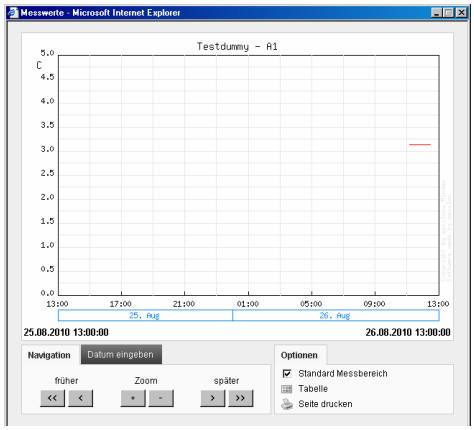


Select the device with “MyDevice”.ID

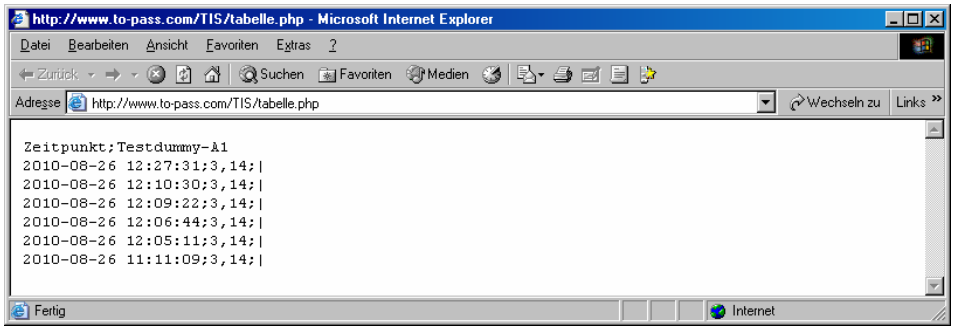


Press “Display” button to show last logged values.

Further, you can click on Icon  to open logged data as a trend chart.



Link table (Tabelle) displays CSV data.



Please note WAGO provides “WagoLibToPass_02.lib” a specialized WAGO-TO-PASS library covering the whole WAGO-TO-PASS portal functionality.

Example_04: WAGO-TO-PASS multiple request

As given by the WAGO-TO-PASS specification, only 8 analog input and 8 digital input values can be stored via one HTTP-POST-Request to the server.

To store wider input set the PLC should process multiple HTTP-POST-Request to the server, i.e. the PLC should work like to be a group of virtual WAGO-TO-PASS clients running independently and communicating to the server simultaneously. Each virtual client is related to one set of input variables.

The **WagoLibHttp02_Example04.pro** shows how to implement such solution. It shows both how to log data to server and how to read setpoints received from server.

The project consists of 3 units

- “PLC_PRG” assigns values to be logged to the server (“InitDeviceIds” action) and sets client identifiers (“AssignProcessData” action). Further, it calls “TOPASS_clientMux”
- “TOPASS_clientMux” is a for-loop based process, executing each client using the “TOPASS_client” program. Data to be transmitted are collected in “astServerRequestData”. Setpoints and state of communication process of each device are collected to the “astServerResponseData” array. Possible valid data from server is announced via “xResponseDataAvailable”. Success, warnings and possible communication errors are announced via block output counters. Cyclic behavior can be optionally set by “xCyclic” flag.
- “TOPASS_client” is the one using the HTTP POST functionality. It builds server HTTP request body and process the request to server.

More precisely, an example of the HTTP-POST-Request to WAGO TO-PASS-Portal is as follows

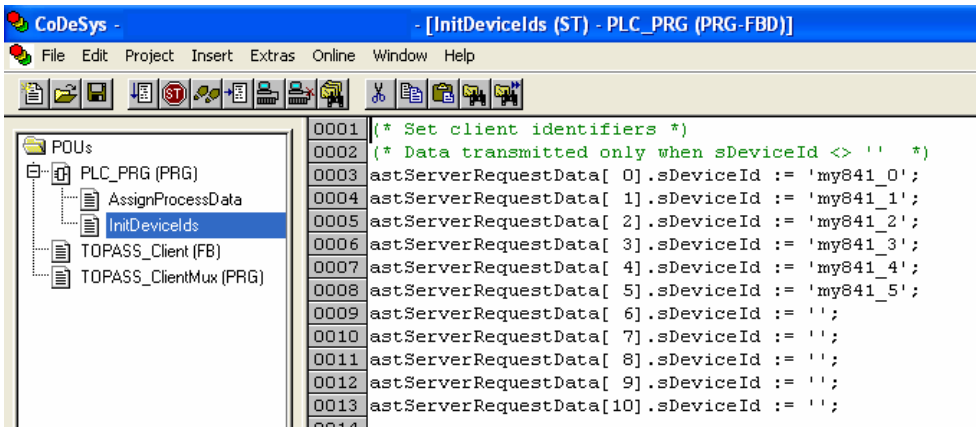
```
POST /wago/savedata.php HTTP/1.1
Host: www.to-pass.com
User-Agent: WAGO 750-841
Connection: Keep-Alive
Content-TYPE: application/x-www-form-urlencoded
Content-length: 251

ID=MyDevice;12345&PA=0060;home;99&TI=10/08/25,17:30:49
&D1=1;0&D2=0;1&D3=1;0&D4=0;0&D5=1;1&D6=0;0&D7=1;0&D8=1;1
&A1=3.14;.C;0&A2=4711;%;1&A3=237.2;Volt;0&A4=3189;kWh;0
&A5=652.1;Watt;1&A6=50.2;Umin;0&A7=14.8;Amper;0&A8=4.3;qm/h;1
&MV=1234;5678;90AB;CDEF;AFFE
```

Request of each client has to be identified by a unique PLC ID identifier parameter. For example

```
ID=my841_1
ID=my841_2
ID=my841_3
...
```

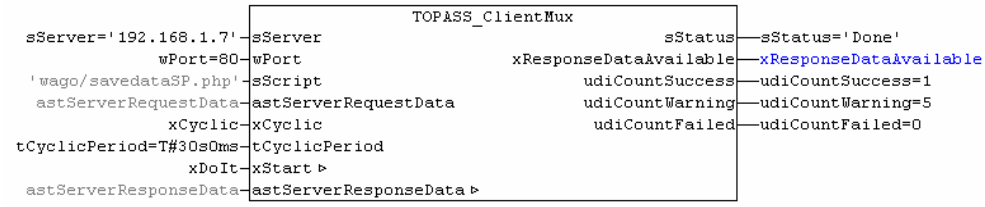
Note: to configure your own device Ids, contact Wago support department (support@wago.com) to configure it properly on the wago-to-pass server.

As described, “Id” of each client can be configured in the “InitDeviceIds” action of PLC_PRG.



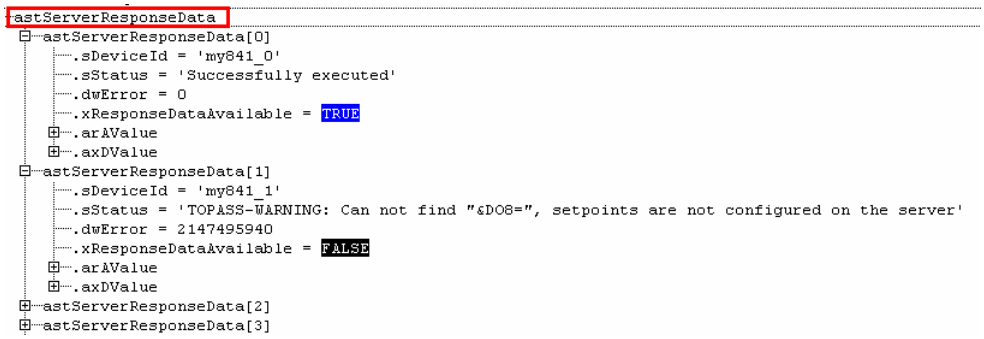
Each client has its own data set prepared in the “AssignProcessData” action of “PLC_PRG”.

The multi WAGO-TO-PASS client process is executed by “xDoIt” variable in the PLC_PRG

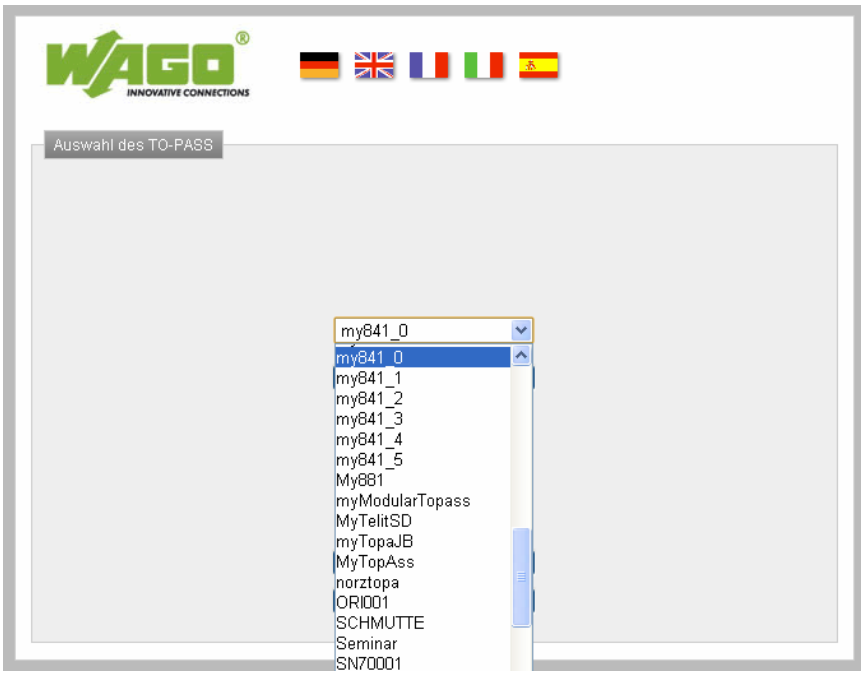


The status of the process is presented via status and error output variables.

Output values and each client status information, received from server, are stored into “astServerResponseData” structure. Incurable errors and warnings have string and DWORD interpretation there.



To see the logged data on the http://www.to-pass.com server, login as the “test“ user (user name: test, password: test). Select the corresponding device ID from the device selector, e.g. my841_0, and click “Display” button.



Data of the selected device are presented in the “Digital input” and “Analog input” fields. Setpoints can be set in the “Setpoints” area.

Example_05: Own HTTP-API using PHP

This example shows how to create your own customer specific web API using PHP scripting language. The example API works as follows:

The HTTP GET request

```
http://<host_name>/test/my_http_api.php?param=<get_param>
```

Where <host_name> is name of server to process the service, e.g. localhost or 127.0.0.1 if server installed locally. The <value> is a string to be presented in response of the server.

The HTTP response is a string

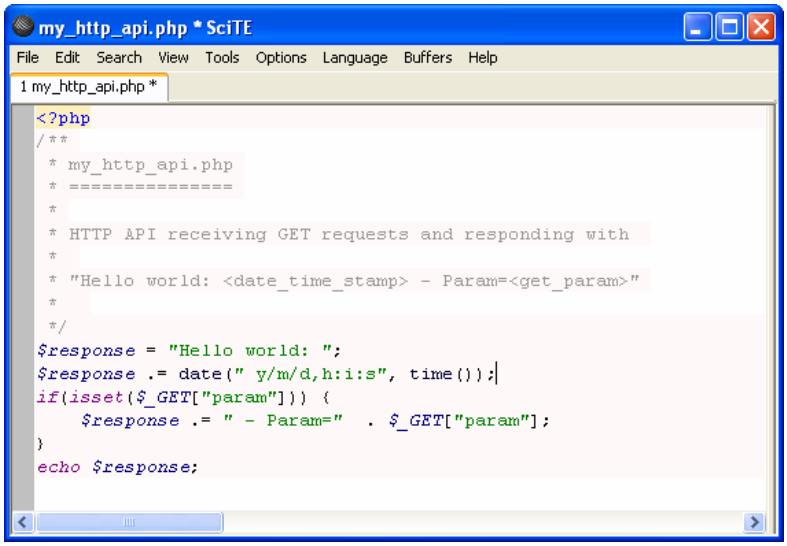
```
Hello word <date_time_stamp> - Param=<get_param>
```

Where <date_time_stamp> is script execution date stamp and <get_param> is the string received in the GET request

Note this example requires a running web server with enabled PHP engine. For the local testing the XAMPP test environment is recommended, for more see “Annex A: Test web server environment setup” chapter.

HTTP server settings

Open your web server file system using FTP or a file browser. Create folder “/test” in the web server document root directory (“~/xampp/htdocs/” for local XAMPP solution). Copy “my_http_api.php” into “/test”

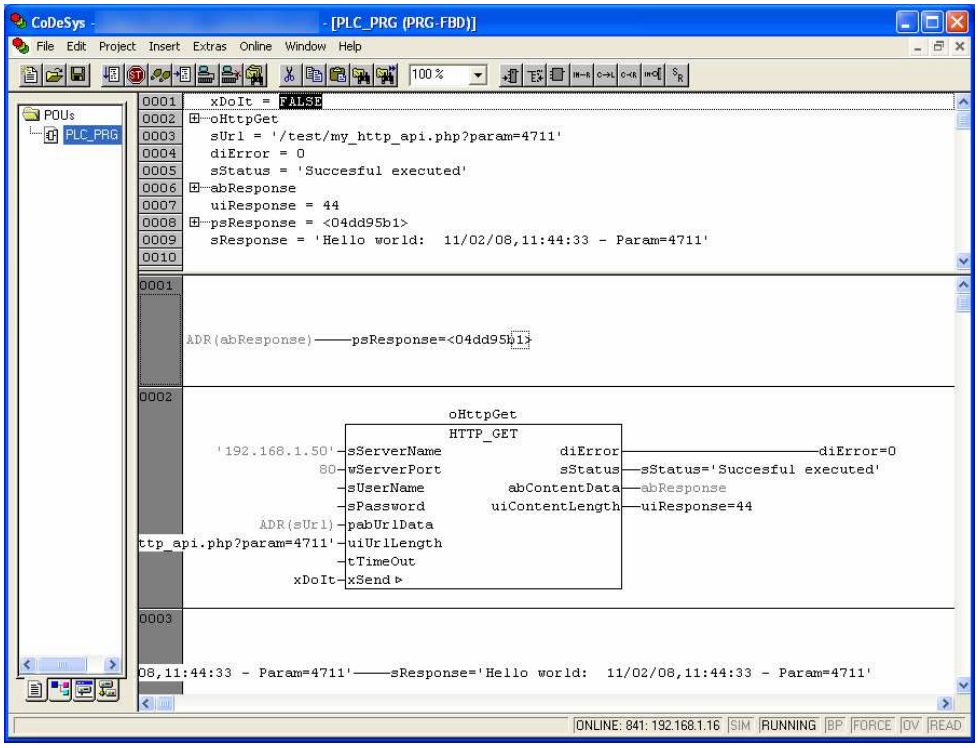


Test the script functionality with standard browser.

PLC program

Open **WagoLibHttp02_Example05.pro**, set servers <host_name> to "sServerName"and set variable "sUrl" for example to

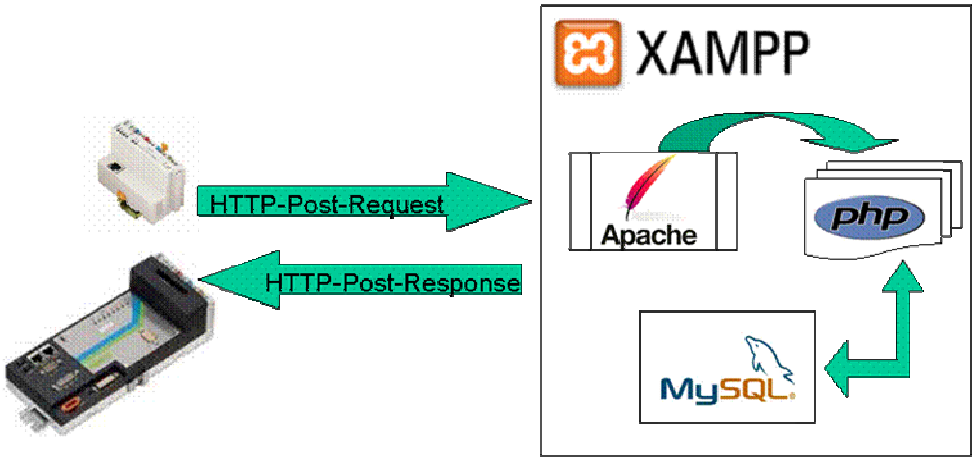
```
/test/my_http_api.php?param=1234
```



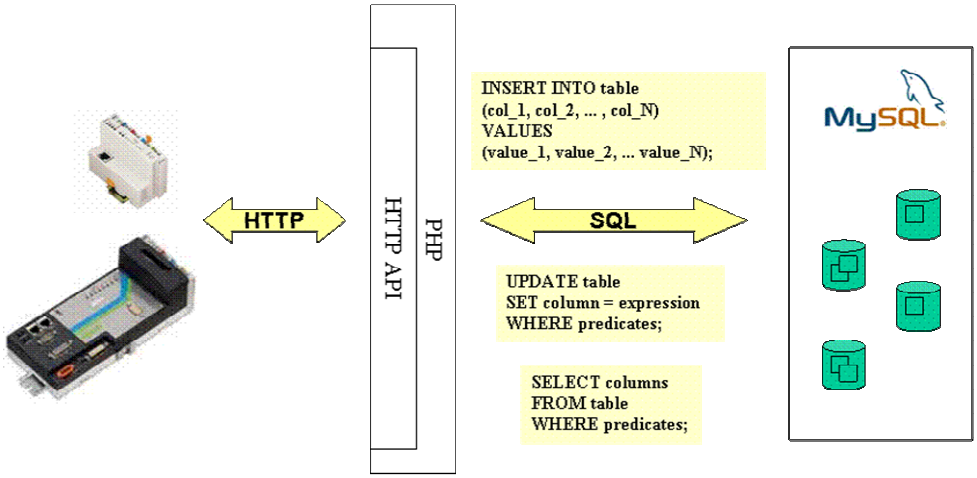
Start the code using "xDoIt", variable "sResponse" will show the response from server.

Example_06: Database access via HTTP-API

This example aim is to show the interaction between a PLC program and a database engine (in this case MySQL) to store a process data, to log an alarms and to read a data from such database based web portal and use them in controller application.



The PLC-Program sends a HTTP request together with some parameters to a Web-Server. The Web-Server forwards the request parameter to PHP engine and process the PHP script. In this example the scripts transact with the MySQL database engine and insert the parameters into a database table or read data from the database.

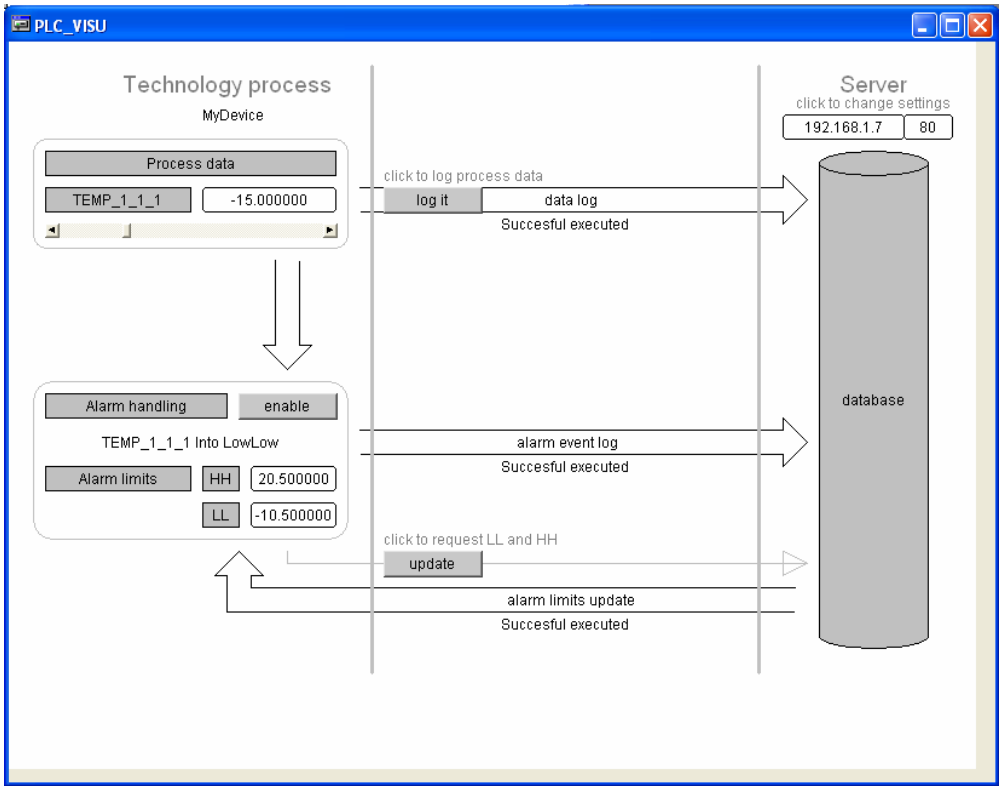


Please note this example requires a running HTTP web server, the PHP engine and the MySQL database server. For the test purposes the local installation of the XAMPP solution is recommended, for more see “Annex A: Test web server environment setup” chapter.

PLC program and program setup

Open **WagoLibHttp02_Example06.pro** in CodeSys 2.3. The Example functionality is as follows:

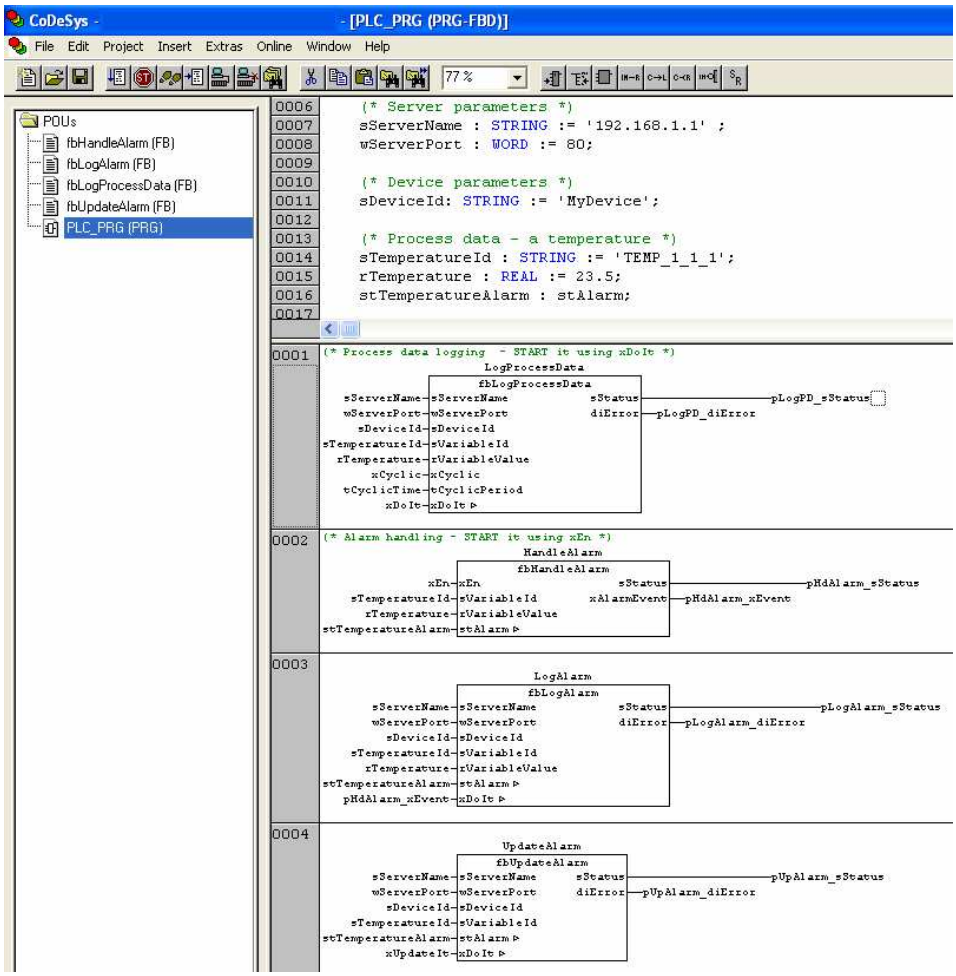
- Logs process data from a technology to a server database, i.e. temperature identified. Data are logged on demand.
- If enabled, it sends process data alarms to the server, i.e. when an alarm event occurs than device id, variable id, group, event type, limits (LowLow, HighHigh) and a short message are sent to the server.
- Allows changing the alarm limits either locally or from the server database.



The PLC program consists of four units.

- "LogProcessData" logs temperature process data. Data parameters are converted to the HTTP GET URL request and sent to the portal database via API in "process_data_api.php" script.
- "HandleAlarm" manages alarm event process based on state of temperature. It controls the "LogAlarm".
- "LogAlarm" logs alarm parameters to the server. Alarm data parameters are converted to the HTTP GET request and sent to the portal database via API implemented in "alarm_api.php" PHP script.

- "UpdateAlarm" is used to update LowLow and HighHigh alarm parameters from server database on demand. The update API is provided by "alarm_update_api.php" script. Further the POU decodes the server response and set the alarm parameters.



To set the program working well, set the server name and the port correctly ("sServerName" and "wServer Port"). For server hostnames don't forget to configure the PLC DNS server setting for host name resolution.

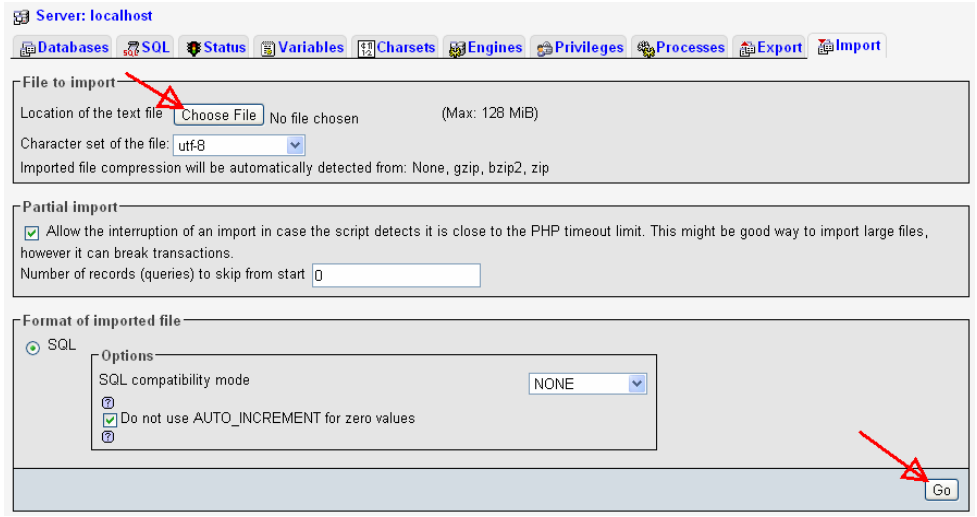
Database "test" setup

To setup the example database use a MySQL administration tool, e.g. **PhpMyAdmin** that is a part of XAMPP package (To open local tool use http://localhost or http://127.0.0.1 and click the "PhpMyAdmin**"** link in the main XAMPP menu).

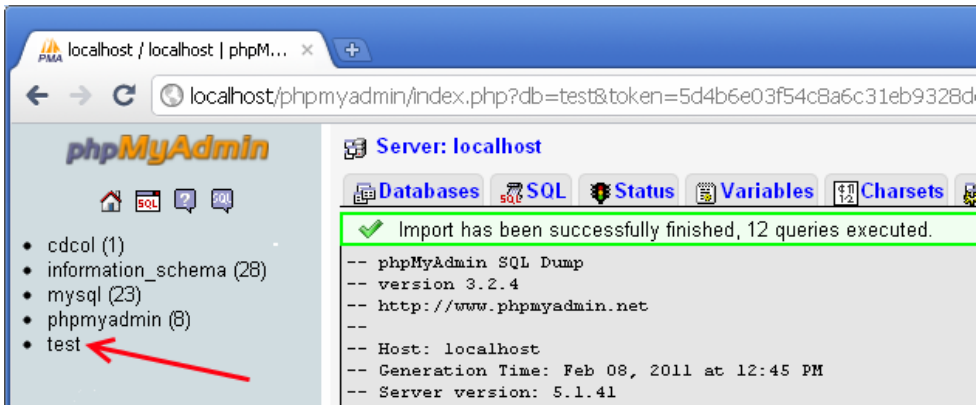
This example database "test" includes 3 tables. To create the database structure import attached SQL batch file **test_db_and_tables.sql**. (when the your "test" database is occupied use "test_tables_only.sql").

To import the SQL batch file use the "Import" tab of the tool

Further choose the SQL batch file and press Go

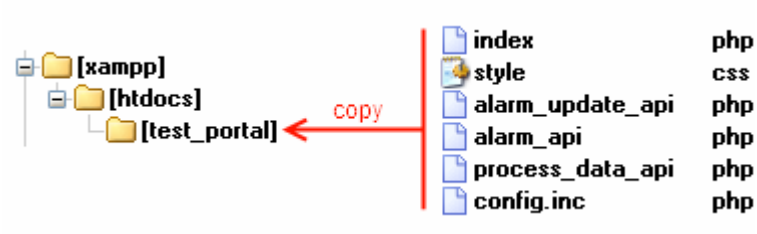


Tool will announce you the import result. After success the “test” database item is added into database list. Click on it to see the database structure.



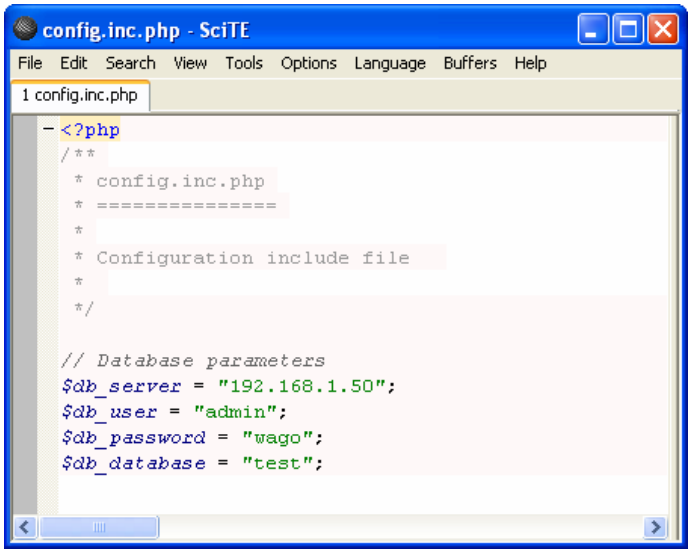
PHP script setup and publishing

Create the folder “test_portal” in the web server’s root folder “~/xampp/htdocs” and copy all example PHP scripts there



Filename:	Description:
index.php	Example for a PHP representation layer to show stored data.
process_data_api.php	HTTP API for process data
alarm_api.php	HTTP API for alarms
alarm_update_api.php	HTTP API for alarm update
config.inc.php	Database constant definition
style.css	Cascading Style Sheet for representation layer

To make changes to the default server values (server host name, MySQL user name, MySQL password, database name, etc.), for that open and edit the “config.inc.php” script using a text editor (e.g. Notepad).



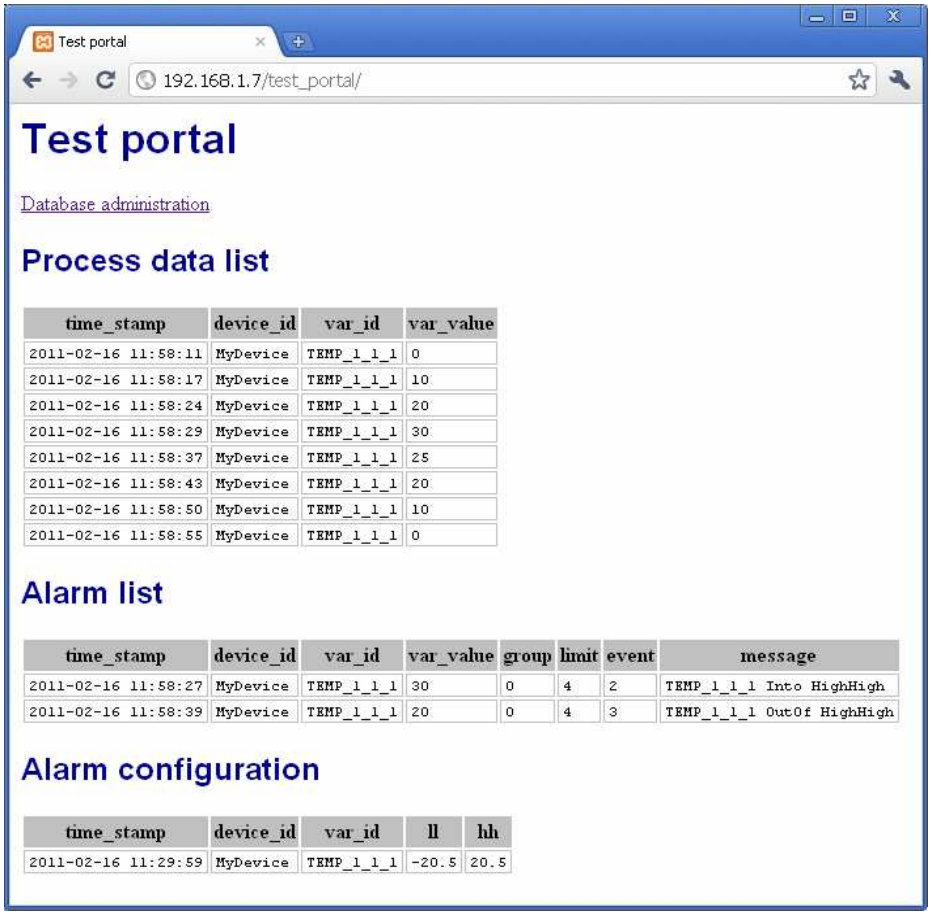
The server is now configured and ready to use.

Test and result analyze

The example provides a basic representation layer as a starting point for developing your own web portal. Using the address

```
http://<server_host_name>/test_portal/index.php
```

(For example http://127.0.0.1/test_portal/index.php for the server installed locally) the portal data will be presented as follows.



Now run the PLC application and test to log process data and alarms.

For a deeper insight on how it all works, the debugging tool “wireshark” may be helpful.

Additional information

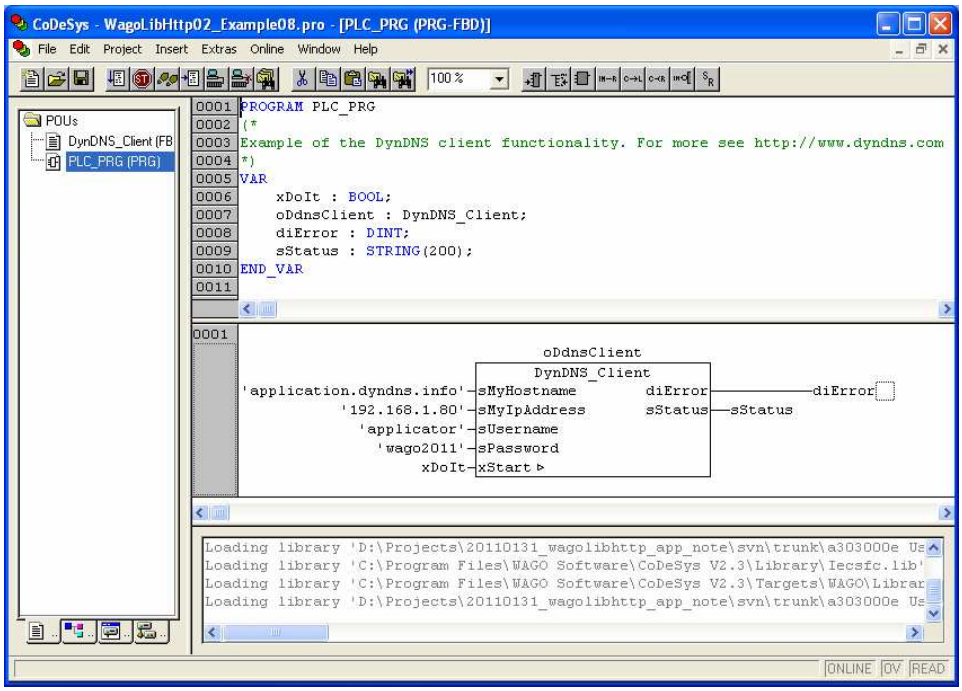
To see a web portal offering more functionality search for WAGO-TO-PASS web portal solution. Under http://www.to-pass.com/wago-demo/ you will find WAGO example for the representation layer of a heating system (login as user “wago” with password “demo”).

In a case you don’t want to develop your own representation layer scripts contact info@wago.com for more information.

Example_07: Dynamic DNS service DynDNS

Dynamic DNS is a network service that provides the capability for a networked device to notify a Domain Name System (DNS) server to change the active DNS configuration of its configured hostnames, IP addresses or other information in real time. When a change in IP address is found or a PLC alters any of their settings, the client should perform an update. For that special HTTP request should be sent to the server.

This example presents the DynDNS service (http://www.dyndns.com) use. To start the example program set the “xDoIt” to TRUE. The result will be presented in the “sStatus” variable.



For more see the provider page, especially the DynDNS HTTP API documentation at http://www.dyndns.com/developers/specs/syntax.html .

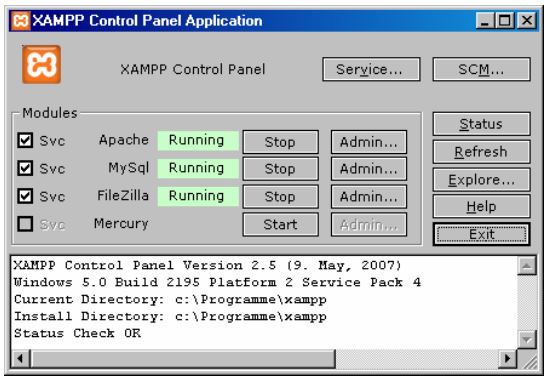
Annex A: Test web server environment setup

The easiest way to get a working web server, PHP-Engine and MySQL-database is installing XAMPP.

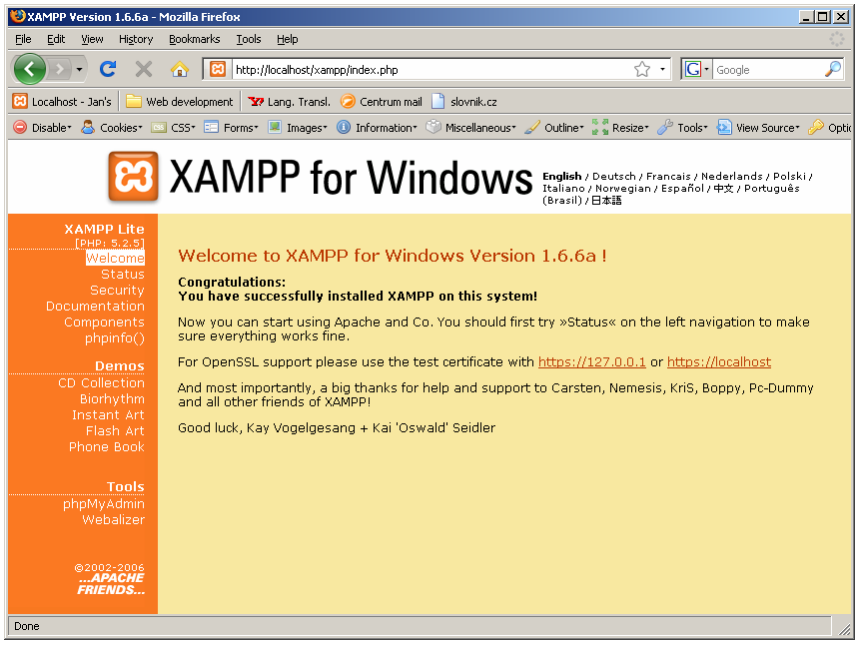
Download the XAMPP package for your operating system from http://www.apachefriends.org/en/xampp.html and follow the installation instructions. This application note is created on a Microsoft operating system.

Please keep in mind that XAMPP provides an easy to install system for developing and testing of web solutions. Care should be taken when applying security settings for your own application.

After installing run “XAMPP Control Panel Application” and check if all services are “Running”.



Open the URL http://localhost or http://127.0.0.1 and examine the XAMPP examples and tools.



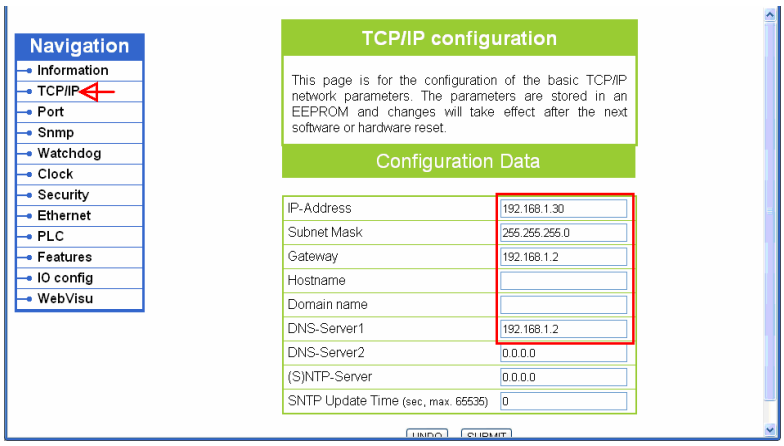
Now the web server, PHP-Engine and MySQL-Database are ready to use.

Annex B: Network problem analysis

If some Internet services are not reachable the problem could be wrong environment setting. Check following items

- *PLC and router reachability*. Check if both PLC and router are reachable using **ping** command. In PLC code you can use the "SOCK_PING" function from the "WagoLibSockets.lib".

- *PLC Ethernet settings.* Use either the "Wago Ethernet settings" tool or the PLC "Wago Web based Management" (WBM) and check the PLC TCP/IP settings. Set reachable IP-Address, subnet mask, Gateway and the DNS server if needed. After setting restart the PLC.



- *DNS server reachability*. Check the DNS router is available using for example "SysSockGetHostByName" function from "SysLibSockets.lib".

- *Router, firewall, NAT etc. settings*. Please check settings of all participated network devices. For example, wrong setting of firewall or router can block whole communication.

- *Network analysis.* For deeper network analysis use the Wireshark http://www.wireshark.org/ and a net sniffer. For example the approved net sniffer is the Wago 852-104, i.e. manageable Ethernet switch with port mirroring functionality.

.

WAGO Kontakttechnik GmbH & Co. KG
Postfach 2880 • D-32385 Minden
Hansastraße 27 • D-32423 Minden
Telefon: 05 71/8 87 – 0
Telefax: 05 71/8 87 – 1 69
E-Mail: info@wago.com

Internet: http://www.wago.com