Module <HTTP> of subsystem “Protocols”

Module:	HTTP
Name:	HTTP
Type:	Protocol
Source:	prot_HTTP.so
Version:	1.5.0
Author:	Roman Savochenko
Translated:	Maxim Lysenko
Description:	Provides support for the HTTP protocol for WWW-based user interfaces.
License:	GPL

Contents table

Introduction

Module of the transport protocol HTTP is designed to support the implementation of network protocol HTTP (Hypertext Transfer Protocol) in the system OpenSCADA.

HTTP Protocol is used to transfer the WWW contents. For example, via HTTP the following types of documents are transmitted: html, xhtml, png, java, and many others. Adding the HTTP support in OpenSCADA system together with the Sockets transport allows to implement various user functions based on the WWW interface. The module implements two main methods of the HTTP protocol: "GET" and "POST". "HTTP" module provides control of the integrity of HTTP-queries and, jointly with "Sockets" transport, allows to "collect" holistic requests of their fragments, as well as maintain the keeping of the connection alive (Keep-Alive).

For flexible connection of the user interfaces to the module the modular mechanism within the module HTTP is used. In the role of modules the modules of subsystem the "User interfaces" are used with the additional information field "SubType" with the value of "WWW".

In the requests for the Web resources the URL(Universal Resource Locator) are commonly used, hence the URL is passed as the main parameter via HTTP. The first element of the requested URL is used to identify the module UI. For example URL: http://localhost:10002/WebCfg means - address to module WebCfg on the host http://localhost:10002. In the case of an incorrect indication of the module ID, or when you address without identifier of the module at all, HTTP module generates the dialogue of the information on the input and with the choice of one of the available user interfaces. Example of a dialogue is shown in Figure 1.

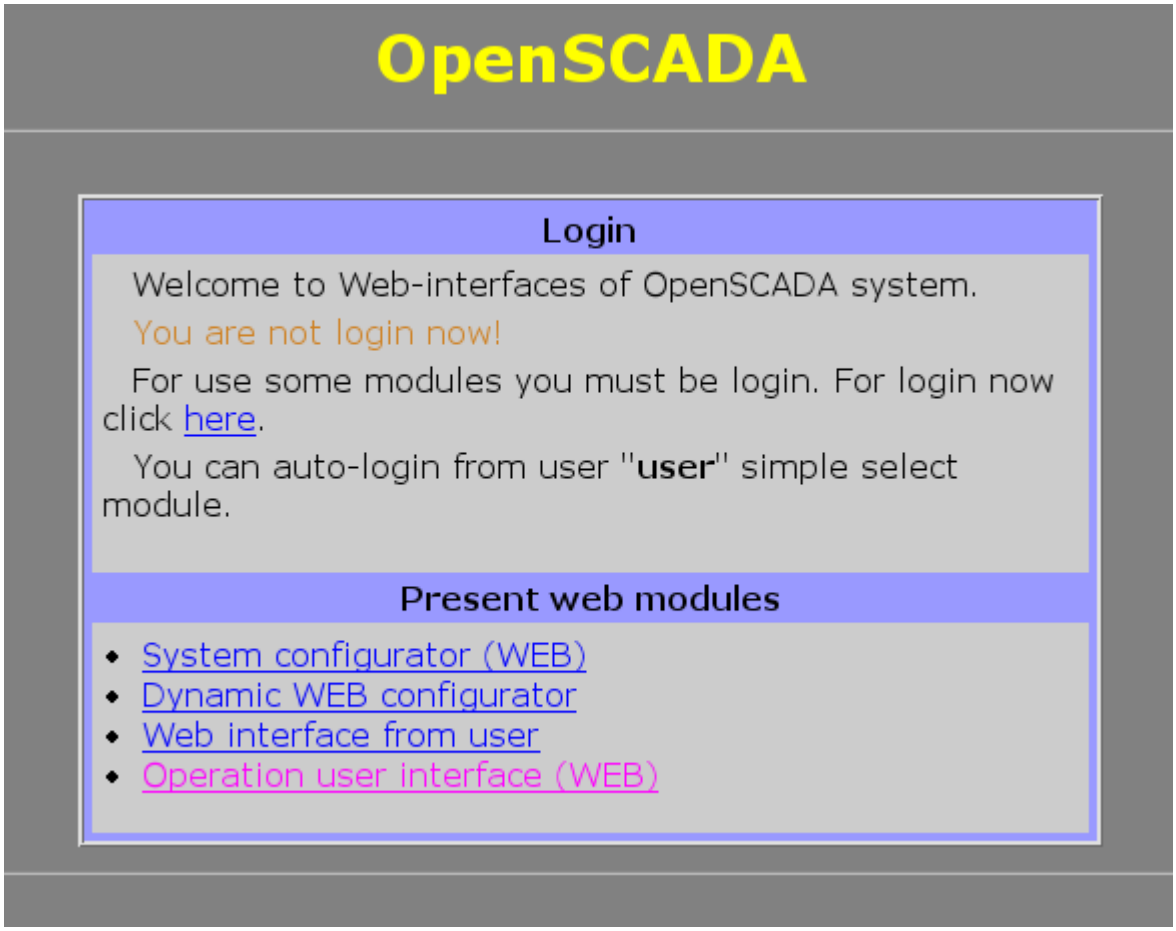

Fig.1. Dialog of the choice of WWW-interface module.

1. Authentication

Module supports authentication in the system OpenSCADA while providing access to the WEB-interface modules (Fig.2). Dialogue is formed in the language of XHTML 1.0 Transitional!

Fig.2. Authentication dialogue in the system OpenSCADA.

For ease of Web-based interface module provides the ability to automatically log on behalf of the specified user. Configuring automatic logon to make by the module settings page (Fig.3).

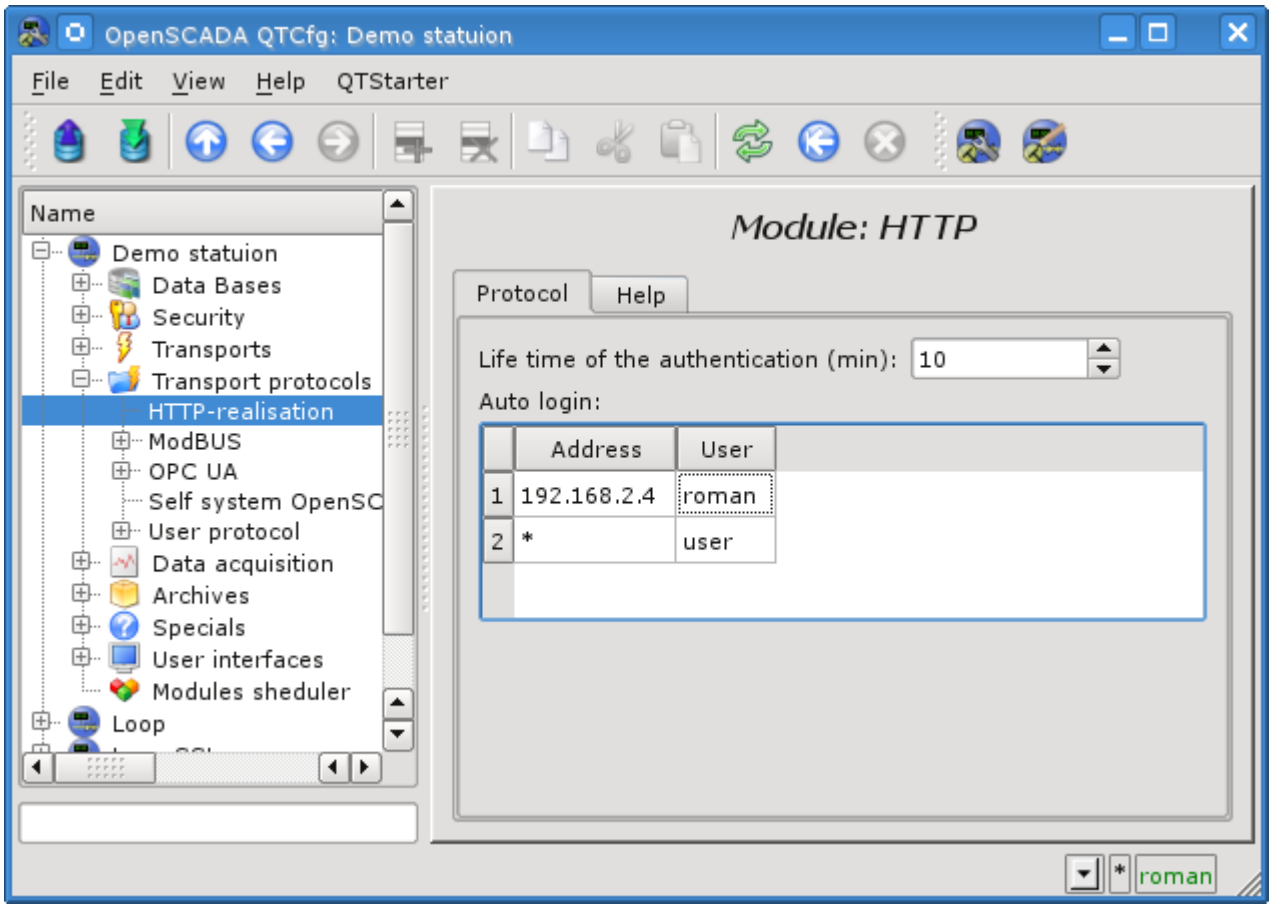

Fig.3. The module configuring page.

On the module settings you can specify the lifetime of the authentication and set up automatic login. Automatic login is carried out by matching the address indicated in the column "Address", on behalf of the user specified in the column "User".

2. The modules of user WEB-interface

Modules of the user interface (UI) designed to work with HTTP module, should indicate an information field "SubType" with the value "WWW" and "Auth" field with the value "1" if the module requires an authentication at login. For communication of HTTP module and UI modules an advanced communication mechanism is used. This mechanism involves the export of interface functions. In this case the UI modules must export the following function:

- *void HttpGet(const string &url, string &page, const string &sender, vector<string> &vars, const string &user);* - GET method with the parameters:
  url - address of the request; *page* - page with the answer; *sender* - address of the sender; *vars* - request variables; *user* - user of the system.
- *void HttpPost(const string &url, string &page, const string &sender, vector<string> &vars, const string &user);* - POST method with the parameters:
  url – address of the request; *page* – page with the answer and with the contents of the body of the POST request; *sender* – address of the sender; *vars* – request variables; *user* - user of the system.

Then, in the case of a HTTP GET request, the function HttpGet will be called, and in the case of the POST request, the function HttpPost will be called in the appropriate UI module.

3. Outgoing requests function's API

The outgoing function of API operate by HTTP-request's content which wrapped to XML-packages. The request structure is:

```
<req Host="host" URI="uri">
      <prm id="pId">pVal</prm>
      <cnt name="cName" filename="cFileName">
            <prmid="cpId">cpVal</prm>
            cVal
      </cnt>
</req>
```

Where:

- *req* - request method, supported methods "GET" and "POST".
- *host* - http-server address into format *[HostAddr]:[HostIp]*. If that field have been passed then used node address which set into address field of the transport.
- *uri* - resource address, file or direcory, at http-server.
- *pId*, *pVal* - identifier and value of addition http-parameters. You can set multiply http-parameters by different *prm* tags set.
- *cName*, *cFileName*, *cVal* - name, file-name and value of content-element of POST-request. You can set multiply content-elements by different *cnt* tags set.
- *cpId*, *cpVal* - identifier and value of addition content-parameters. You can set multiply content-parameters by different *prm* tags set.

Request result's structure is:

```
<req Host="host" URI="uri" err="err" Protocol="prt" RezCod="rCod" RezStr="rStr">
      <prm id="pId">pVal</prm>
      respVal
</req>
```

Where:

- *req* - request method.
- *host* - http-server address.
- *uri* - resource address.
- *err* - the error wich appear in request time. For successed requests the field is empty.
- *RezCod*, *RezStr* - request result into view code and text.
- *pId*, *pVal* - identifier and value of addition http-parameters. Respond can set multiply http-parameters by different *prm* tags set.
- *respVal* - respond's content.

Into example role we accord using the function into users procedures for GET and POST requests making by language JavaLikeCalc.JavaScript:

```
//GET request
req = SYS.XMLNode("GET");
req.setAttr("URI","/");
SYS.Transport.Sockets.out_testHTTP.messIO(req,"HTTP");
test = req.text();

//POST request
req = SYS.XMLNode("POST");
req.setAttr("URI","/WebUser/FlowTec.txt");
cntNode = req.childAdd("cnt").setAttr("name","pole0").setAttr("filename","Object2-
k001-100309-17.txt");
cntNode.childAdd("prm").setAttr("id","Content-Type").setText("text/plain");
cntText = "Object2-k001\r\n";
cntText += "\r\n";
cntText += "v002\r\n";
```

```
cntText += " n1\r\n";
cntText += "  09.03.10 16   Polnyj    7155.25    216.0  32.000  17.5\r\n";
cntText += "v005\r\n";
cntText += " n1\r\n";
cntText += "  09.03.10 16   Polnyj     188.81    350.0   4.000  40.0\r\n";
cntText += "\r\n";
cntNode.setText(cntText);
SYS.Transport.Sockets.out_testHTTP.messIO(req,"HTTP");
```