

Модуль подсистемы “Протоколы” <HTTP>

Модуль:	HTTP
Имя:	HTTP
Тип:	Протокол
Источник:	prot_HTTP.so
Версия:	1.5.0
Автор:	Роман Савоченко
Описание:	Предоставляет поддержку протокола HTTP для WWW-основанных пользовательских интерфейсов.
Лицензия:	GPL

Оглавление

Модуль подсистемы “Протоколы” <HTTP>.....	1
Введение.....	1
1. Аутентификация.....	2
2. Модули пользовательского WEB-интерфейса.....	3
3. API функции исходящих запросов.....	4

Введение

Модуль транспортного протокола HTTP предназначен для реализации поддержки сетевого протокола HTTP(Hypertext Transfer Protocol) в системе OpenSCADA.

Протокол HTTP используется для передачи содержимого WWW. Так, через HTTP передаются следующие типы документов: html, xhtml, png, java и многие другие. Добавление поддержки HTTP в систему OpenSCADA в комплексе с транспортом Sockets позволяет реализовывать различные пользовательские функции на основе WWW интерфейса. Модуль HTTP реализует два основных метода протокола HTTP: GET и POST. Модуль HTTP обеспечивает контроль целостности HTTP-запросов и, совместно с транспортом Sockets, позволяет “собирать” целостные запросы из их фрагментов, а также обеспечивать сохранение соединения живым (Keep-Alive).

Для гибкого подключения пользовательских интерфейсов к данному модулю используется модульный механизм в рамках самого модуля "HTTP". В роли модулей используются модули подсистемы “Пользовательские интерфейсы” с дополнительным информационным полем “SubType”, имеющим значение “WWW”.

В запросах к Web ресурсам принято использовать URL(Universal Resource Locator), следовательно URL передаётся как основной параметр через HTTP. Первый элемент запрашиваемого URL используется для идентификации модуля UI. Например URL: http://localhost:10002/WebCfg означает обращение к модулю "WebCfg" на хосте http://localhost:10002. В случае ошибочного указания идентификатора модуля или при обращении вообще без идентификатора "HTTP" модуль генерирует диалог информации о входе и с предоставлением выбора одного из доступных пользовательских интерфейсов. Пример диалога показан на рисунке 1.

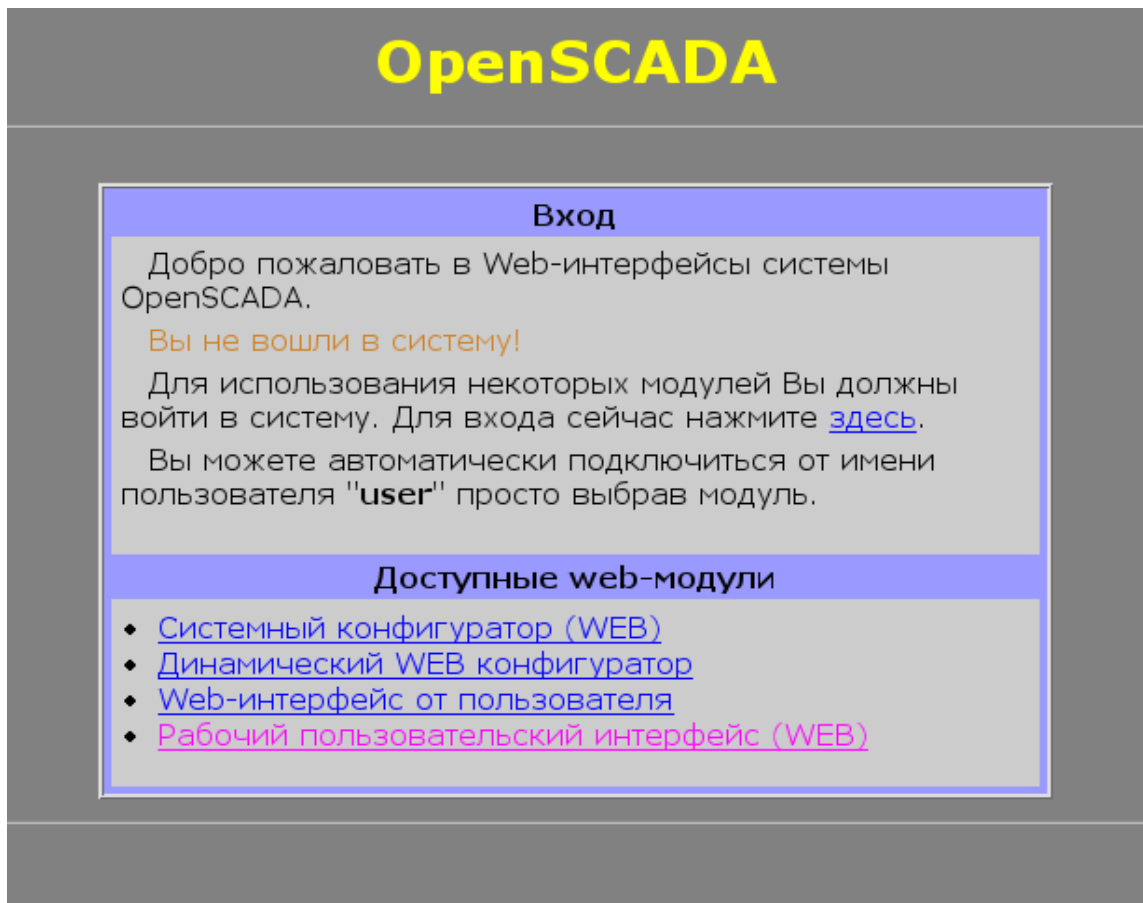


Рис.1. Диалог выбора модуля WWW-интерфейса.

1. Аутентификация

Модулем поддерживается аутентификация в системе OpenSCADA при предоставлении доступа к модулям WEB-интерфейсов (рис.2). Диалог формируется на языке XHTML 1.0 Transitional!

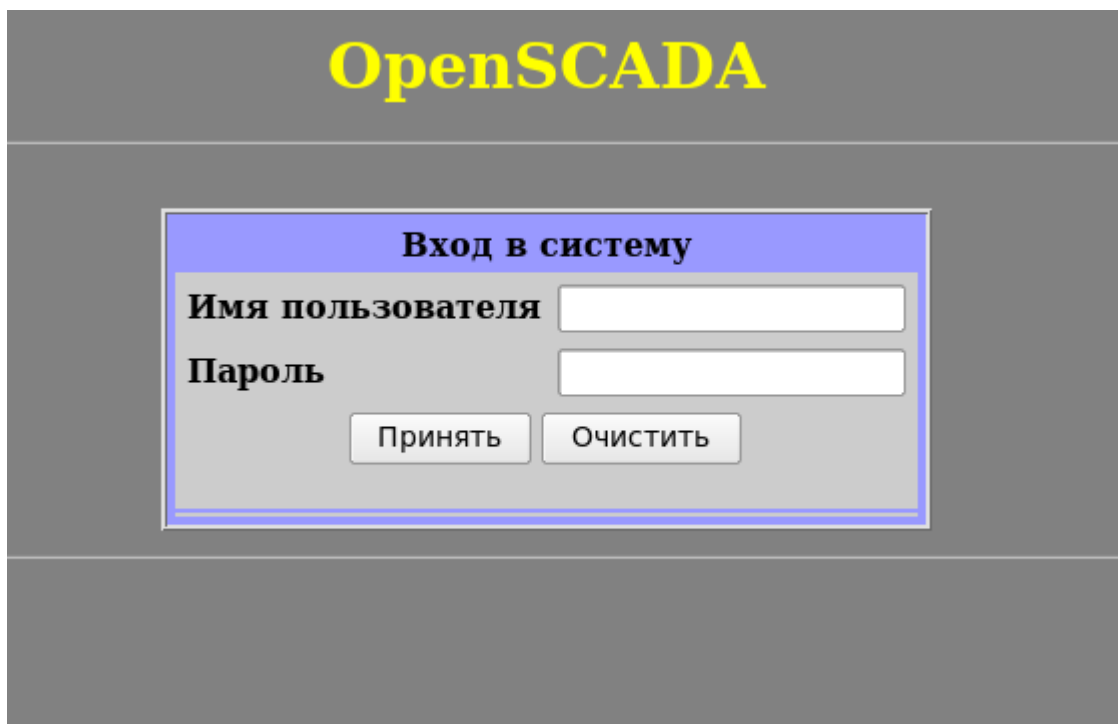


Рис.2. Диалог авторизации в системе OpenSCADA.

Для облегчения работы с Web-интерфейсами модуль предусматривает возможность автоматического входа от имени указанного пользователя. Конфигурация автоматического входа осуществляется на странице настройки модуля (рис.3).

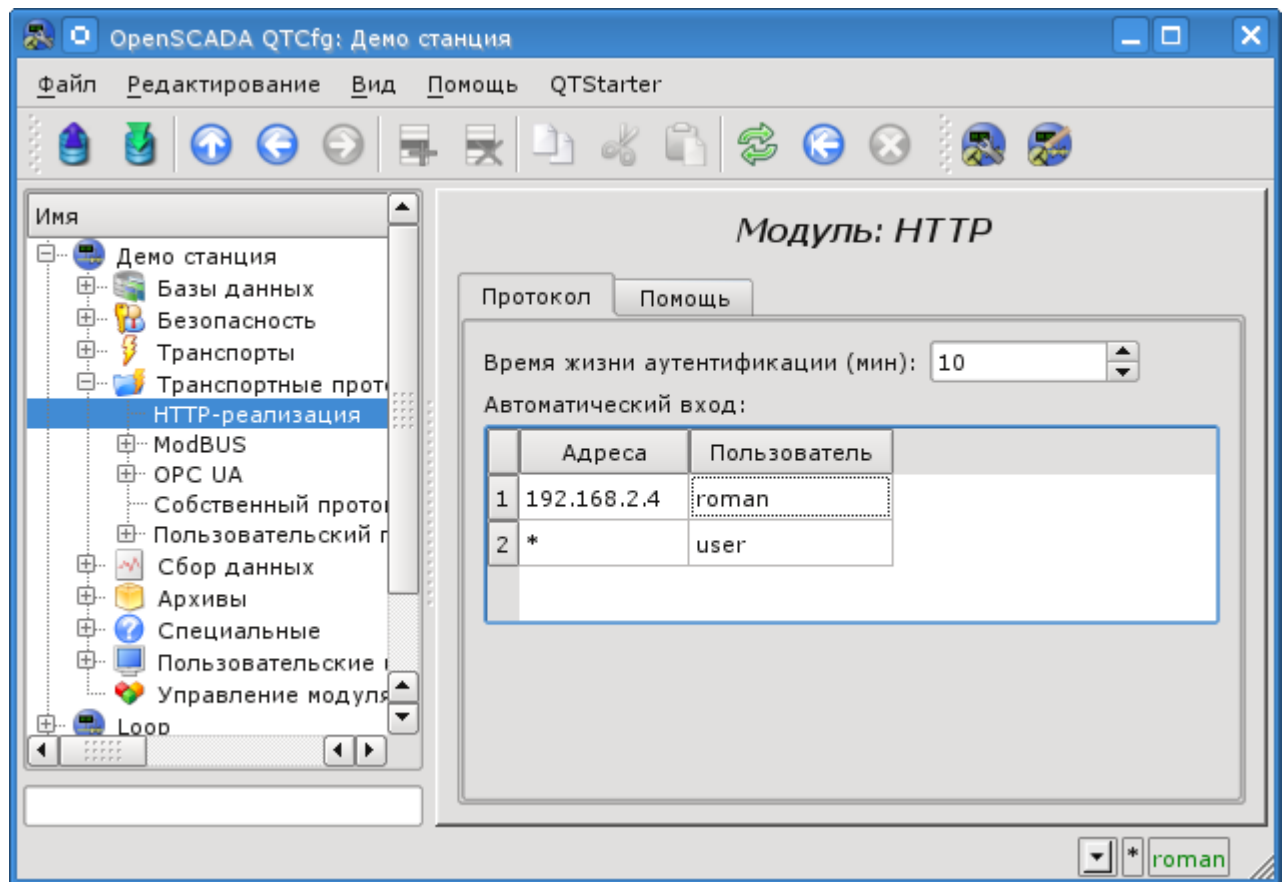


Рис.3. Страница настройки модуля.

На странице настройки модуля можно указать время жизни аутентификации и настроить автоматический вход. Автоматический вход осуществляется по совпадению адреса, указанного в колонке "Адреса", от имени пользователя, указанного в колонке "Пользователь".

2. Модули пользовательского WEB-интерфейса

Модули пользовательского интерфейса (UI), предназначенные для совместной работы с модулем HTTP, должны устанавливать информационное поле "SubType" значением "WWW" и поле "Auth" значением "1", если модуль требует аутентификации при входе. Для связи модуля HTTP и модулей UI используется расширенный механизм коммуникации. Этот механизм подразумевает экспорт интерфейсных функций. В данном случае UI модули должны экспортировать функции:

- `void HttpGet(const string &url, string &page, const string &sender, vector<string> &vars, const string &user);` - Метод GET с параметрами:
`url` - адрес запроса; `page` - страница с ответом; `sender` - адрес отправителя; `vars` - переменные запроса; `user` - пользователь системы.
- `void HttpPost(const string &url, string &page, const string &sender, vector<string> &vars, const string &user);` - метод POST с параметрами:
`url` - адрес запроса; `page` - страница с ответом и содержимым тела запроса POST; `sender` - адрес отправителя; `vars` - переменные запроса; `user` - пользователь системы.

Далее в случае поступления HTTP запроса "GET" будет вызываться функция "HttpGet", а в случае запроса "POST" будет вызываться функция "HttpPost" в соответствующем модуле UI.

3. API функции исходящих запросов

Функция исходящих запросов оперирует обменом содержимым HTTP-запросов, завернутыми в XML-пакеты. Структура запроса имеет вид:

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

Где:

- *req* - метод запроса, поддерживаются методы "GET" и "POST".
- *host* - адрес узла http-сервера в формате *[HostAddr]:[HostIp]*. Если данное поле опущено то используется адрес узла, указанный в поле адреса транспорта.
- *uri* - адрес ресурса, обычно файл или директория, на http-сервере.
- *pId*, *pVal* - идентификатор и значение дополнительного http-параметра. Http-параметров может быть указано множество, отдельными *prm* тегами.
- *cName*, *cFileName*, *cVal* - имя, имя-файла и значение элемента содержимого POST-запроса. Элементов содержимого может быть указано множество, отдельными *cnt* тегами.
- *cpId*, *cpVal* - идентификатор и значение дополнительного параметра содержимого. Параметров содержимого может быть указано множество, отдельными *prm* тегами.

Результатом запроса является ответ со структурой:

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

Где:

- *req* - метод запроса.
- *host* - адрес узла http-сервера.
- *uri* - адрес ресурса.
- *err* - ошибка, возникшая во время запроса. В случае успешного запроса это поле пустое.
- *RezCod*, *RezStr* - результат запроса в виде кода и текста.
- *pId*, *pVal* - идентификатор и значение дополнительного http-параметра, ответа. Http-параметров может быть множество, определённый отдельными *prm* тегами.
- *respVal* - содержимое ответа.

В качестве примера использования данной функции в пользовательских процедурах приведём формирование GET и POST запросов на языке JavaLikeCalc.JavaScript:

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

//POST запрос
req = SYS.XMLNode("POST");
req.setAttr("URI", "/WebUser/FlowTec.txt");
cntNode
req.addChild("cnt").setAttr("name", "pole0").setAttr("filename", "Object2-k001-100309-17.txt");
cntNode.addChild("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");
```