HttpCwsServerExample

README

Last Updated August 8th, 2019

Contents

[Overview 3](#_Toc16152924)

[REST API Overview 4](#_Toc16152925)

[URL Routing Table (starting from /cws/api) 4](#_Toc16152926)

[HTTP Verb Table 5](#_Toc16152927)

[Console Commands 6](#_Toc16152928)

[StartServer 6](#_Toc16152929)

[RlyOpen 6](#_Toc16152930)

[RlyClose 6](#_Toc16152931)

[Equipment 7](#_Toc16152932)

[Devices 7](#_Toc16152933)

[Software / Firmware 7](#_Toc16152934)

[Important Notes 7](#_Toc16152935)

[Before Loading the Program 7](#_Toc16152936)

[Loading the Program and Project Files 8](#_Toc16152937)

# Overview

This HttpCwsServer project was created to exercise the HttpCwsServer class and related types from the SIMPL# Pro library and to instruct readers with annotated source code how these types can be applied in an application.

In this example, an HttpCwsServer allows the control system to expose a public REST API for the control system’s set of relay ports. This allows remote HTTP clients not only to query the current state of any relay port on the control system, but also to open or close the relay via an HTTP PUT request. In addition, this example program allows clients to *subscribe* either to an individual relay or to the entire relay collection—the server accepts subscription requests containing a callback URL provided by the client; later, the server may dispatch a POST request representing a notification to the callback URL whenever the subscribed-to relay’s state changes. This is an implementation of the [RESTful WebHooks](https://webhooks.pbworks.com/w/page/13385128/RESTful%20WebHooks) standard, and many clients can receive notifications about various relay ports on the control system in this manner.

**Note:** To fully exercise the capabilities of the HttpCwsServerExample program, the user’s HTTP client must be capable of:

* Making HTTP GET, HEAD, PUT, and POST requests for querying, updating, and subscribing to relays
* Listening for POST requests to a callback URL to receive state-change notifications from the relays that the client has subscribed to

This means that a standard web browser would only allow the user to see the current state of the relays through GET requests, but not to modify them or subscribe to them. As a result, the reader is encouraged to use a more fully-featured API testing application such as Postman while experimenting with this project. In addition, the reader could also load the HttpClientExample program to a separate control system. The HttpClientExample program is designed to consume the HttpCwsServerExample’s REST API, and thus it can also be used to observe the behavior of the program.

All SIMPL# Pro source code defining the program is provided in the following files:

* ControlSystem.cs
* RestfulRelayServer.cs

These source files are thoroughly commented and are intended to demonstrate one of many possible applications of the SIMPL# Pro APIs.

## REST API Overview

This example program uses an HttpCwsServer object to provide a REST API that allows clients to manage the control system’s relays. HTTP clients make GET and PUT requests to access and change the relays’ states. Of course, clients cannot DELETE relay ports or POST/PUT more of them.

Furthermore, both the relay collection and each individual relay have an associated subscription URL, or “WebHook URL.”  POSTing to these URLs with a single callback URL in the request body causes a subscription to be registered with an obscure ID, which is designed to prevent other users from deleting the subscription resource. This subscription URL is linked in the response so that the client may DELETE the subscription object later on, thereby unsubscribing. HttpCwsServerExample handles these POST requests by creating a Subscription object with a SubscriberCallbackUrl instance variable, which refers to the URL the client provided in the body of the POST request. This notification process is an implementation of the informal “RESTful WebHooks” standard, which can be found [here](https://webhooks.pbworks.com/w/page/13385128/RESTful%20WebHooks).

The *media type* used to convey the representation of the relays in the HTTP response body is [Collection+JSON](http://amundsen.com/media-types/collection/), whose specification can be found [here](https://github.com/collection-json/spec). Collection+JSON imposes a set of constraints on vanilla JSON to streamline the representation of collections of resources. These constraints work nicely for the purposes of this example project, since it serves a collection of relays, and it has allowed the author of this example program to avoid inventing a new JSON pattern specifically for this project.

Clients can interact with the REST API through the URLs given in the URL Routing Table combined with the HTTP verbs indicated in the HTTP Verb Table. Both of these tables are given below:

### URL Routing Table (starting from /cws/api)

/relays

/relays/web-hooks

/relays/web-hooks/{subid}

/relays/{id}

/relays/{id}/web-hooks

/relays/{id}/web-hooks/{subid}

### HTTP Verb Table

### 

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
|  | /relays | /relays/web-hooks | /relays/web-hooks/{subid} | relays/{id} | relays/{id}/web-hooks | /relays/{id}/web-hooks/{subid} |
| GET | Get relay collection |  |  | Get representation of specific relay |  |  |
| PUT |  |  |  | Update relay state and dispatch notifications to all subscribers to that relay. The request body contains a Collection+JSON template indicating the new state. |  |  |
| POST |  | Subscribe client to the entire relay collection. The request’s entity-body contains a link to the client’s listener URL |  |  | Subscribe client to a relay. The request’s entity-body contains a link to the client’s listener URL |  |
| DELETE |  |  | Unsubscribe from the /relays resource |  |  | Unsubscribe from the /relays/{id} resource |
| HEAD | Get headers, but no body. The response shall include a Link header to /relays/web-hooks |  |  | Get headers, but no body. The response shall include a Link header to /relays/{id}/web-hooks |  |  |

Each populated cell in the HTTP Verb Table is handled by the ProcessRequest method of a class that implements the IHttpCwsHandler interface, corresponding to the URL route. Readers should refer to the RestfulRelayServer.cs source file for the route handlers’ full implementation details with informative comments. Cells that are not populated will return a 405 “Method Not Allowed” response.

## Console Commands

The HttpCwsServer program provides three console commands:

### StartServer

*startserver*

This console command requires no arguments and simply starts the CWS server. Once started, clients may, for example, navigate to http://<control-system-hostname-or-ip>/cws/api/relays to retrieve a Collection+JSON representation of the collection of relays.

### RlyOpen

*rlyopen <index>*

Open the specified relay. The index provided represents the ID of the relay port to be opened. For example, if the control system running the program has 4 relay ports, the IDs will be numbered 1 to 4. When invoked, WebHooks notifications will dispatch to any subscribers to the specified port. If indicated relay port is already opened, this command has no effect.

### RlyClose

*rlyclose <index>*

Close the specified relay. The index provided represents the ID of the relay port to be closed. For example, if the control system running the program has 8 relay ports, the IDs will be numbered 1 to 8. When invoked, WebHooks notifications will dispatch to any subscribers to the specified port. If indicated relay port is already closed, this command has no effect.

# Equipment

This program is designed to work with the following hardware/software:

## Devices

* 3-Series Control System
* A separate HTTP client, such as a Postman client or the HttpClientExample project

## Software / Firmware

|  |  |
| --- | --- |
| Device | Firmware Version |
| 3-Series Control System | 1.601.xxx or later |

|  |  |
| --- | --- |
| Software | Version |
| Toolbox | 3.03.xxx or later |

# Important Notes

## Before Loading the Program

* SSL must be turned **off** to run this program—Once registered, the HttpCwsServer object will serve responses using HTTP or HTTPS depending on the SSL setting. However, WebHooks notifications are sent using the HttpClient class, which requires SSL to be disabled to function.

# Loading the Program and Project Files

1. Open the top-level solution file, HttpCwsServerExample.sln, in Visual Studio 2008.
2. Press F6 to Build All. This will place the .cpz file in the HttpCwsServerExample\bin\Debug folder.
3. Transfer the .cpz file to one of the control system’s program slots.
4. On the control system, invoke **progload -p:#**, where **#** is the number of the slot into which you placed the .cpz file.