HTTPS Client V3.0
HTTP Client V3.0
SIMPL Windows Application Guide

Description

This SIMPL windows package is an update to the HTTPS modules previously offered. This Version 3 update offers a significant upgrade to the previous modules with the modules functionally able to accomplish the same task, however in this version, with many more features and capabilities allowing for greater flexibility.

This SIMPL Windows module package allows SIMPL Windows programs to send configurable HTTP and HTTPS requests. Programmers can populate the request parameters in the module parameters or override them using digital, serial, and analog inputs, making the modules suitable for systems driven by configuration files.

The modules support common request methods (such as GET, POST, and more), ability to send custom headers, request bodies, and both Basic and Digest authentication with automatic handling of Digest challenges. The HTTPS module adds optional control over SSL certificate verification. Server responses are returned via a serial output along with the response code. Errors are logged to the processor's error log for troubleshooting, and live debugging is available through console debug output.

Supported Processors

Any 3 or 4 series appliance, or VC-4 instance, with Ethernet. Internet access is not required. This module is not supported on 2-series or earlier processors.

Compatibility			Processor Requirements	
4 SERIES CONTROL COMPATIBLE	3 COMPATIBLE	NOT COMPATIBLE SERIES	Ethernet REQUIRED	REMOVABLE MEDIA NOT REQUIRED

Contents

Module Application

Both modules include parameters for IP address or hostname, port, username, password, request type, content string, additional headers, and enabling basic authentication. At program start, the module uses the values entered into the parameters to construct requests. Requests are constructed and entered into a FIFO Queue on change of the `request_path$`.

After program start, if any parameter values are updated via the accompanying override analog or serial inputs (such as when loading a system from a configuration), the module will use the overridden values for all subsequent requests.

Requests are constructed using the configured IP address or hostname, port, request type (GET, POST, PUT, DELETE, etc.), request path, content string (request body), and any additional headers (comma-delimited). When the `request_path$` input changes, the module builds the full request and sends it to the target endpoint.

For endpoints requiring Basic authentication, the module includes a parameter to enable sending the Authorization: Basic header. When the **`Enable Auth Basic Authentication`** parameter is set to **`Yes`**, the credentials provided in the **`Username`** and **`Password`** parameters (or overridden via the `[username_override$]` and `[password_override$]` inputs) are Base64-encoded and sent in the Basic authentication header.

For endpoints requiring Digest authentication, if the server responds indicating that Digest authentication is required, the module follows the Digest authentication workflow and automatically retries the request using the credentials provided in the **`Username`** and **`Password`** parameters (or their override inputs), along with the values supplied in the server's Digest challenge. At this time, Digest authentication is supported only using the MD5 algorithm.

Once the Digest parameters are received, the module caches these values and attempts to reuse them for subsequent requests, eliminating the need to receive a new challenge each time. The Digest cache expires after five minutes, at which point the next request proceeds normally and triggers a new Digest challenge, restarting the cycle. Changing the target IP address or hostname, or the port, clears the Digest cache. This approach improves performance by eliminating approximately half of the authentication challenge round-trips.

If a request completes successfully, the response body returned by the server is output on the `[response$]` serial signal, and the associated HTTP response code is output on the `[response_code]` analog output signal.

If a request fails for any reason (connection failure, authentication error, protocol error, or internal error), an error entry is written to the processor's error log to assist with troubleshooting.

For HTTPS communication, SSL certificate verification may be enabled or disabled via parameters or override inputs. Disabling certificate verification allows communication with self-signed or internal certificates.

When `[enable_debug]` is latched high, the module outputs detailed diagnostic information to the console, including request construction details, headers, authentication state, and server responses. Latching this signal low disables all debug output. Debug mode should remain disabled during normal operation.

Module License

This Module is covered under our One-Time Purchase/Download Product (Type 1) terms. The full terms are available in the final section of this help file.

Here are some common questions we receive for this license type:

Q. Do I have to pay a license each time I use the module?
A. No. Your company pays a single license fee and may use the module as many times as needed in systems programmed by your company.

Q. Does the module need a license key or contact an activation server to be licensed.
A. No. No keys or activation servers. Just the honor system. Please follow the rules and use it respectfully. (We reserve the right to change this in the future.)

Q. I received this module as part of a project archive or take-over project. Do I need to pay a license fee?
A. Yes. Receiving the module in a project archive or take-over does not grant a license. The module is licensed only to the company that purchased it. The license is non-transferable, and redistribution is not permitted. Your company must purchase its own license before using the module in any systems it programs or maintains; including systems acquired through a take-over.

Q. Can I provide my CSP, subcontractor, or third-party programmer the module under my company's license?
A. No. Redistribution is not permitted. Only the company that purchased the license is authorized to use the module. Third-party programmers, including CSPs, must obtain their own license. If you later take over programming from a CSP, your company will also need to purchase a license.

Q. I have multiple programmers within my company. Can they all use the module?
A. Yes, all programmers working for the licensed company may use the module without additional fees, as long as the module is only used in systems programmed by that same company.

Q. Do I need to purchase updates?
A. No. Updates are distributed through the web store and anyone who has a valid license can download the updates at no charge.

Q. Are there limits on how long the module can be used in a system?
A. No. Once purchased, the license is perpetual and may be used indefinitely in systems programmed by your company.

Signal and Parameter Descriptions
HTTP_Client_V3.0

Bracketed signals such as "[signal_name]" are optional signals

DIGITAL INPUTS

[enable_debug] .....Latch high to print debug messages to the console. Latch low to disable debug messages.

ANALOG INPUTS

[http_port_override].....Analog input that, when changed, overrides the configured **HTTP port** parameter. Valid values are 1d–65535d. The module will use the overridden value for subsequent requests until another override is applied. If the signal is commented out, the module will always use the value defined in the **HTTP Port** Parameter.

[request_type_override] .....Analog input that, when changed, overrides the configured **Request Type** parameter. Valid values are: 0d = GET, 1d = POST, 2d = PUT, 3d = DELETE, 4d = PATCH, 5d = HEAD. The module will use the overridden value for subsequent requests until another override is applied. If the signal is commented out, the module will always use the value defined in the **Request Type** Parameter.

[authentication_basic_override] .....Analog input that, when changed, overrides the configured **Enable Auth Basic Authentication** parameter. Valid values are: 0d = No (disabled), 1d = Yes (enabled). The module will use the overridden value for subsequent requests until another override is applied. If the signal is commented out, the module will always use the value defined in the **Enable Auth Basic Authentication** Parameter.

SERIAL INPUTS

request_path$ .....Whenever the request_path$ input changes, the module constructs a new request using the configured parameter or override values and adds it to the FIFO request queue.

[ip_address_or_hostname_override$].....String input that, when changed, overrides the configured **IP address or hostname** parameter. The module will use the overridden value for subsequent requests until another override is applied. If the signal is commented out, the module will always use the value defined in the **IP address or hostname** parameter.

[username_override$] .....String input that, when changed, overrides the configured U**sername** parameter. The module will use the overridden value for subsequent requests until

another override is applied. If the signal is commented out, the module will always use the value defined in the U**sername** parameter.

[password_override$].....String input that, when changed, overrides the configured **Password** parameter. The module will use the overridden value for subsequent requests until another override is applied. If the signal is commented out, the module will always use the value defined in the **Password** parameter.

[content_string_override$].....String input that, when changed, overrides the configured **Content String** parameter. The module will use the overridden value for subsequent requests until another override is applied. If the signal is commented out, the module will always use the value defined in the **Content String** parameter.

[additional_headers_override$] .....String input that, when changed, overrides the configured **Additional Headers** parameter. The module will use the overridden value for subsequent requests until another override is applied. If the signal is commented out, the module will always use the value defined in the **Additional Headers** parameter. Each header should be comma delimited, with the last item not containing a comma i.e. "`X-Ignore-Me: yes,X-NonFunctional-Test: 12345`".

DIGITAL OUTPUTS

This module does not use any analog outputs.

ANALOG OUTPUTS

[response_code] .....Optional analog output indicating the server's HTTP response code. Common response codes include 200d (OK), 201d (Created), 204d (No Content), 400d (Bad Request), 401d (Unauthorized), 403d (Forbidden), 404d (Not Found), and 500d (Internal Server Error).

SERIAL OUTPUTS

[response$].....Optional serial output containing the body of the response from the server.

PARAMETERS

IP Address.....Enter the default IP Address or hostname of the server the request should be sent to. Can be overridden by initializing `[ip_address_or_hostname_override$]`.

Port .....Enter the default Port number the request should be sent on. Typically this will be 80d for HTTP. Can be overridden by initializing `[http_port_override]`.

Username .....Enter the default User Name that will be sent in requests if the Enable Basic/Digest Authentication parameter is enabled. Can be overridden by initializing `[username_override$]`.

Password .....Enter the default Password that will be sent in requests if the Enable Basic/Digest Authentication parameter is enabled. Can be overridden by initializing `[password_override$]`.

Request Type.....Enter the default Request Type that will be sent for requests. Can be overridden by initializing `[request_type_override].`

Content String .....Enter the default Content String (Request Body of the message) that will be sent for requests. Can be overridden by initializing `[content_string_override$]`.

Additional Headers .....Enter default Headers that will be sent for requests. Each header should be comma delimited, with the last item not containing a comma i.e. "`X-Ignore-Me: yes,X-NonFunctional-Test: 12345`". Can be overridden by initializing [additional_headers_override$].

Enable Auth Basic Authentication .....Select to enable or disable authentication for requests. When enabled, the module attempts the request using the credentials provided in the **Username** and **Password** parameters, or, if overridden via [username_override$] and [password_override$], using the override values, with Basic Authorization via the Authorization: Basic header.

Signal and Parameter Descriptions
HTTPS_Client_V3.0

Bracketed signals such as "[signal_name]" are optional signals

DIGITAL INPUTS

[enable_debug] .....Latch high to print debug messages to the console. Latch low to disable debug messages.

ANALOG INPUTS

[https_port_override] .....Analog input that, when changed, overrides the configured **HTTPS port** parameter. Valid values are 1d–65535d. The module will use the overridden value for subsequent requests until another override is applied. If the signal is commented out, the module will always use the value defined in the **HTTPS Port** Parameter.

[request_type_override] .....Analog input that, when changed, overrides the configured **Request Type** parameter. Valid values are: 0d = GET, 1d = POST, 2d = PUT, 3d = DELETE, 4d = PATCH, 5d = HEAD. The module will use the overridden value for subsequent requests until another override is applied. If the signal is commented out, the module will always use the value defined in the **Request Type** Parameter.

[authentication_basic_override] .....Analog input that, when changed, overrides the configured **Enable Auth Basic Authentication** parameter. Valid values are: 0d = No (disabled), 1d = Yes (enabled). The module will use the overridden value for subsequent requests until another override is applied. If the signal is commented out, the module will always use the value defined in the **Enable Auth Basic Authentication** Parameter.

[ssl_verification_override] .....Analog input that, when changed, overrides the configured **Enable SSL Verification** parameter. Valid values are: 0d = No (disabled), 1d = Yes (enabled). The module will use the overridden value for subsequent requests until another override is applied. If the signal is commented out, the module will always use the value defined in the **Enable SSL Verification** Parameter.

SERIAL INPUTS

request_path$ .....Whenever the `request_path$` input changes, the module constructs a new request using the configured parameter or override values and adds it to the FIFO request queue.

[ip_address_or_hostname_override$].....String input that, when changed, overrides the configured **IP address or hostname** parameter. The module will use the overridden value for subsequent requests until another override is applied. If the signal

is commented out, the module will always use the value defined in the **IP address or hostname** parameter.

[username_override$] .....String input that, when changed, overrides the configured U**sername** parameter. The module will use the overridden value for subsequent requests until another override is applied. If the signal is commented out, the module will always use the value defined in the U**sername** parameter.

[password_override$].....String input that, when changed, overrides the configured **Password** parameter. The module will use the overridden value for subsequent requests until another override is applied. If the signal is commented out, the module will always use the value defined in the **Password** parameter.

[content_string_override$].....String input that, when changed, overrides the configured **Content String** parameter. The module will use the overridden value for subsequent requests until another override is applied. If the signal is commented out, the module will always use the value defined in the **Content String** parameter.

[additional_headers_override$] .....String input that, when changed, overrides the configured **Additional Headers** parameter. The module will use the overridden value for subsequent requests until another override is applied. If the signal is commented out, the module will always use the value defined in the **Additional Headers** parameter. Each header should be comma delimited, with the last item not containing a comma i.e. "`X-Ignore-Me: yes,X-NonFunctional-Test: 12345`".

DIGITAL OUTPUTS

This module does not use any analog outputs.

ANALOG OUTPUTS

[response_code] .....Optional analog output indicating the server's HTTP response code. Common response codes include 200d (OK), 201d (Created), 204d (No Content), 400d (Bad Request), 401d (Unauthorized), 403d (Forbidden), 404d (Not Found), and 500d (Internal Server Error).

SERIAL OUTPUTS

[response$].....Optional serial output containing the body of the response from the server.

PARAMETERS

IP Address.....Enter the default IP Address or hostname of the server the request should be sent to. Can be overridden by initializing `[ip_address_or_hostname_override$]`.

Port .....Enter the default Port number the request should be sent on. Typically this will be 80d for HTTP. Can be overridden by initializing `[http_port_override]`.

Username .....Enter the default User Name that will be sent in requests if the Enable Basic/Digest Authentication parameter is enabled. Can be overridden by initializing `[username_override$]`.

Password .....Enter the default Password that will be sent in requests if the Enable Basic/Digest Authentication parameter is enabled. Can be overridden by initializing `[password_override$]`.

Request Type.....Enter the default Request Type that will be sent for requests. Can be overridden by initializing `[request_type_override]`.

Content String .....Enter the default Content String (Request Body of the message) that will be sent for requests. Can be overridden by initializing `[content_string_override$]`.

Additional Headers .....Enter default Headers that will be sent for requests. Each header should be comma delimited, with the last item not containing a comma i.e. "`X-Ignore-Me: yes,X-NonFunctional-Test: 12345`". Can be overridden by initializing [additional_headers_override$].

Enable Auth Basic Authentication .....Select to enable or disable authentication for requests. When enabled, the module attempts the request using the credentials provided in the **`Username`** and **`Password`** parameters, or, if overridden via `[username_override$]` and `[password_override$]`, using the override values, with Basic Authorization via the Authorization: Basic header.

Enable SSL Verfication .....Select to enable or disable SSL Verification. Valid values are: 0d = No (disabled), 1d = Yes (enabled). Can be overridden by initializing `[ssl_verification_override]`.

Support

This Module is supported by ControlWorks Consulting, LLC. Should you need support for this Module you may email us at support@controlworks.com or call us at:

- (+1) 440 449 1100 (Cleveland, Ohio)
- (+1) 508 695 0188 (Boston, Massachusetts)
- (+1) 202 381 9070 (Washington, DC)
- (+44) (0)20 4520 4600 (London, England)

ControlWorks normal office hours are 9 AM to 5 PM US Eastern time, Monday through Friday, excluding holidays.

Updates

Updates, when available, are free of charge, and are automatically distributed via our webstore. If you have purchased a license, you will receive an email notification to the address entered when the license was purchased. In addition, updates may be obtained using your username and password at https://store.controlworks.com/account/login.aspx.

Distribution Package Contents

The distribution package for this module should include:

HTTPS Client Demo V3.0 (ControlWorks).smw	Demonstration Program
HTTPS_Client_V3.0_(ControlWorks).umc	Main User Module
HTTPS_Client_V3.0_(ControlWorks).usp	SIMPL+ for use inside main module
HTTPS_Client_V3.0_(ControlWorks).ush	SIMPL+ header file, for use inside main module
HTTP_Client_V3.0_(ControlWorks).umc	Main User Module
HTTP_Client_V3.0_(ControlWorks).usp	SIMPL+ for use inside main module
HTTP_Client_V3.0_(ControlWorks).ush	SIMPL+ header file, for use inside main module
HttpClientControlWorks.clz	SIMPL# module for use in SIMPL+ module
HTTPS_Client_V3.0_(ControlWorks)_help.pdf	This help file.

Revision History

V3.0 caleb@controlworks.com 2026.01.24

- Significant modifications and partial rewrite
- Module made dynamic with the addition of:
    - IP Address or Hostname parameter and serial input for overriding the parameter.
    - HTTP(s) Port parameter and analog input for overriding the parameter.
    - Username parameter and serial input for overriding the parameter.
    - Password parameter and serial input for overriding the parameter.
    - RequestType parameter and analog input for overriding the parameter.
        - Support for multiple HTTP methods: GET, POST, PUT, DELETE, PATCH, HEAD
    - Content String parameter and serial input for overriding the parameter.
    - Additional Headers parameter and serial input for overriding the parameter.
        - Headers are comma-delimited, except for the final header.
    - Auth Basic parameter and analog input for overriding the parameter.
        - Enabling sends the credentials provided as a Auth Basic header.
- Request queue enhancements.
- Added Digest authentication cache to improve transaction round-trip performance. When the server supports credential caching, the module reuses cached credentials instead of performing a two-request challenge/response sequence.
- Added support for additional Digest QOP values.
- Digest authentication supports the MD5 algorithm only; SHA-256 and SHA-512 are not supported at this time.

V2.1 caleb@controlworks.com 2025.11.24

- HTTP Module – unreleased.

V2.0 caleb@controlworks.com 2025.11.24

- Release Version

V1.0 caleb@controlworks.com 2020.08.18

- Internal versions.

Development Environment

This module version was developed on the following hardware and software. Different versions of hardware or software may or may not operate properly. If you have questions, please contact us.

Crestron Hardware	**Firmware Version**
CP4	v2.8006.00110
Software	**Software Version**
SIMPL Windows	4.28
Device Database	200.345
Crestron Database	224.05

ControlWorks Consulting, LLC Type 1 Module/Module License Agreement

Definitions:
ControlWorks, *We*, and *Us* refer to ControlWorks Consulting, LLC, with headquarters located at 701 Beta Drive, Suite 22 Mayfield Village, Ohio 44143-2330. *You* and *Dealer* refer to the entity purchasing the module. *Client* and *End User* refer to the person or entity for whom the Crestron hardware is being installed and/or will utilize the installed system. *System* refers to all components described herein as well as other components, services, or utilities required to achieve the functionality described herein. *Module* refers to files required to implement the functionality provided by the module and may include source files with extensions such as UMC, USP, USH, CLZ, SMW and VTP. *Demo Program* refers to a group of files used to demonstrate the capabilities of the Module, for example a SIMPL Windows program and VisionTools Touchpanel file(s) illustrating the use of the Module but not including the Module. *Software* refers to the Module and the Demo Program.

Disclaimer of Warranties
ControlWorks Consulting, LLC software is licensed to You as is. You, the consumer, bear the entire risk relating to the quality and performance of the Software. In no event will ControlWorks Consulting, LLC be liable for direct, indirect, incidental or consequential damages resulting from any defect in the Software, even if ControlWorks Consulting, LLC had reason to know of the possibility of such damage. If the Software proves to have defects, You and not Us must assume the cost of any necessary service or repair resulting from such defects.

Provision of Support
We provide limited levels of technical support only for the most recent version of the Module as determined by Us. We do not provide support for previous version of the module, modifications to the module not made by Us, to persons who have not purchased the module from Us. In addition, we may decline to provide support if the Demo Program has not been utilized. We may withdraw a module from sale and discontinue providing support at any time and for any reason, including, for example, if the equipment for which the Module is written is discontinued or substantially modified. The remainder of your rights and obligations pursuant to this license will not be affected should ControlWorks discontinue support for a module.

Modification of Software
You may not decrypt (if encrypted), reverse engineer, modify, translate, disassemble, or de-compile the Module in whole or part. You may modify the Demo Program. In no event will ControlWorks Consulting, LLC be liable for direct, indirect, incidental or consequential damages resulting from You modifying the Software in any manner.

Indemnification/Hold Harmless
ControlWorks, in its sole and absolute discretion may refuse to provide support for the application of the Module in such a manner that We feel has the potential for property damage, or physical injury to any person. Dealer shall indemnify and hold harmless ControlWorks Consulting LLC, its employees, agents, and owners from any and all liability, including direct, indirect, and consequential damages, including but not limited to personal injury, property damage, or lost profits which may result from the operation of a program containing a ControlWorks Consulting, LLC Module or any component thereof.

License Grant
Software authored by ControlWorks remains the property of ControlWorks. ControlWorks grants You the non-exclusive, non-transferable, perpetual license to use the Software authored by ControlWorks as a component of Systems programmed by You. This Software is the intellectual property of ControlWorks Consulting, LLC and is protected by law, including United States and International copyright laws. This Software and the accompanying license may not be transferred, resold, or assigned to other persons, organizations or other Crestron Dealers via any means.

The use of this software indicates acceptance of the terms of this agreement.