The OPC-HTTP Gateway
PRINTED MANUAL

OPC-HTTP Gateway

Printed: 7/29/2023

Publisher

AGG Software

Production

©2016-2023 AGG Software
http://www.aggsoft.com

Table of Contents

1 About

This program implements a simple OPC-HTTP gateway and allows your web server pages to interact with OPC servers through secure channels. The program acts as an HTTP server and an OPC DA1, DA2, DA3, and UA client at the same time. It translates each HTTP(S) request into a corresponding call to an OPC server.

2 License, registration, and technical support

2.1 License

Copyright © 1999-2023 AGG Software.
All Rights Reserved

SOFTWARE LICENSE

Trial Limited Version

The trial limited version of this software may be used for evaluation purposes at the user's own risk for a trial period. At the end of the trial period, the user must either purchase a license to continue using the software or remove it from his/her system.

The trial limited version may be freely distributed, provided the distribution package is not modified. No person or company may charge a fee for the distribution of OPC-HTTP Gateway without written permission from the copyright holder.

Licensed Version

On payment of the appropriate license fee, the user is granted a non-exclusive license to use OPC-HTTP Gateway on one computer (i.e. a single CPU), for any legal purpose, at a time. The registered software may not be rented or leased but may be permanently transferred, if the person receiving it agrees to the terms of this license. If the software is an update, the transfer must include the update and all previous versions.

Registered customers are entitled to free updates during one year from the date of purchase. It means that for one year you can download and install the latest registered versions of OPC-HTTP Gateway from our site. If you would rather not purchase an update, you can use the program forever; it will never expire, but you won't be able to use the latest version. If you purchased the software more than one year ago, you are no longer entitled to free upgrade and technical support; however, you can purchase an update to the latest version at a special, greatly discounted price, and this update will allow you to have free updates and technical support for another year. The type of the update license must match the type of your existing license.

Whilst every care has been taken in the construction and testing of this software, it is supplied subject to the condition that the user undertakes to evaluate the suitability of the control for his/her purposes. AGG Software makes no representation of the software's suitability for any purpose, and the user agrees that AGG Software has no responsibility for any loss or damage occasioned by the use of this software.

TO THE MAXIMUM EXTENT PERMITTED BY APPLICABLE LAW, THE SOFTWARE, AND DOCUMENTATION ARE PROVIDED "AS IS" AND AGG SOFTWARE DISCLAIMS ALL OTHER WARRANTIES AND CONDITIONS, EITHER EXPRESS OR IMPLIED, INCLUDING, BUT NOT LIMITED TO, IMPLIED WARRANTIES OF MERCHANTABILITY, FITNESS FOR A PARTICULAR PURPOSE, CONFORMANCE WITH DESCRIPTION, TITLE AND NON-INFRINGEMENT OF THIRD-PARTY RIGHTS.

TO THE MAXIMUM EXTENT PERMITTED BY APPLICABLE LAW, IN NO EVENT SHALL AGG SOFTWARE BE LIABLE FOR ANY INDIRECT, INCIDENTAL, CONSEQUENTIAL, SPECIAL OR EXEMPLARY DAMAGES OR LOST PROFITS WHATSOEVER (INCLUDING, WITHOUT LIMITATION, DAMAGES FOR LOSS OF BUSINESS PROFITS, BUSINESS INTERRUPTION, LOSS OF BUSINESS INFORMATION, OR ANY OTHER PECUNIARY LOSS) ARISING OUT OF THE USE OR INABILITY TO USE THE SOFTWARE PRODUCT, EVEN IF AGG SOFTWARE HAS BEEN ADVISED OF THE POSSIBILITY OF SUCH DAMAGES. IN ANY CASE, AGG SOFTWARE'S CUMULATIVE AND ENTIRE LIABILITY TO YOU OR ANY OTHER PARTY FOR ANY LOSS OR DAMAGES RESULTING FROM ANY CLAIMS, DEMANDS OR ACTIONS ARISING OUT OF OR RELATING TO THIS AGREEMENT SHALL NOT EXCEED THE PURCHASE PRICE PAID FOR THIS LICENSE.

Should any term of these terms and conditions be declared void or unenforceable by any court of competent jurisdiction, such declaration shall have no effect on the remaining terms hereof.

If you do not agree to these conditions, you should not install this software.

2.2 Limitations

The unlicensed program works in the trial mode. The program allows testing all features, but it limits the time or amount of processed data. The license key removes all limitations from the trial version. You may purchase a license key here.

The trial version of our software has the following limitations:

- The trial period is limited to 21 days. After that time, the program stops working.
- The continuous program work time is limited to two hours. After that period, the program shows a message and stops working.
- All data export modules can handle the first 100 records only.
- In the spy mode you can receive the first 65535 bytes only.

2.3 How to acquire a license

The unlicensed program works in the trial mode. The license key removes all limitations from the trial version and allows you to use our technical support for one year.

If you want to buy a program through the Internet, visit the order page of our site. On this page, you can get the newest information about the registration process, and also find an order link. Please follow the “Buy now” link, enter your personal information, and choose the most convenient payment method for you. Further, you will get a notification and follow the notes in it.

You may find more information about our policies, payment terms, payment methods, and frequently asked questions on our website.

2.4 Support

Technical questions	support@aggsoft.com
Common questions	info@aggsoft.com
Sales questions	sales@aggsoft.com

3 Installation

3.1 System requirements

Windows 2000 Professional - Windows 10 (2019), including x64 and x86 OS, Workstation, and Server OS.

It is necessary to have at least one free COM port, not busy by any device (mouse, for example) to connect an external device.

3.2 Installation process

If any beta-version was installed on your computer, remove it.

Quit of the working OPC-HTTP Gateway on installation time.

Run an installation file.

By default, the installation wizard installs OPC-HTTP Gateway in the "C:\Programs Files\OPC-HTTP Gateway" or "C:\Programs Files (x86)\OPC-HTTP Gateway" directory of your system disk, but you can change this path.

In the standard distributive of OPC-HTTP Gateway are no additional modules files, which you can download from our site.

4 Configuration

4.1 File with settings

The opchttpgate.xml file contains server configuration and must be placed in the same folder as the opchttpgate.exe file. The OPC UA gateway configuration file has the "opcuahttpgate.xml" name, and the executable name is "opcuahttpgate.exe". You may find some notes about the OPC UA implementation in the following topic.

If you have changed anything in the configuration file, you need to restart the HTTP server.

Example:

```
<Config>
 <Log file="opchttpgate.log" info="1" warning="1" error="1" limit="1000000" />
 <HTTPServer port="80;443" ssl="0" auth="1" staticfilespath="files\" log="1"
logdetailed="0">
  <Users>
   <User login="admin" password="admin" />
   <User login="user" password="user" />
  </Users>
  <AllowedIPs>
   <IP>127.0.0.*</IP>
   <IP>192.168.1.*</IP>
  </AllowedIPs>
 </HTTPServer>
</Config>
```

4.2 Log

The "Log" XML node contains the configuration of the program’s log file. The program can output all messages of a selected type to the specified log file for debugging, security, or process monitoring purposes.

Example:

```
<Log file="opchttpgate.log" info="1" warning="1" error="1" limit="1000000" />
```

file (optional; default: empty value) - The path and name of the log file. If this value is empty, the server will not write anything to the log file.

info (optional; default: 1) - If "1" (true), the program will write all informational messages to the log file.

warning (optional; default: 1) - Same as above for warning messages.

error (optional; default: 1) - Same as above for error messages.

limit (optional; default: 100000) - The log file size limit in bytes.

4.3 Main window

The "DesktopModeSettings" XML node contains the configuration of the main window when the program running in the desktop mode.

Example:

```
<DesktopModeSettings>
  <CustomTitle addversion="1">OPC-HTTP Gateway</CustomTitle>
  <CustomLabel>Uptime</CustomLabel>
  <ExitPassword enabled="1">admin</ExitPassword>
  <StopPassword enabled="1">admin</StopPassword>
</DesktopModeSettings>
```

CustomTitle (optional; default: empty) - The program will place this text in the window caption. If the "addversion" attribute is "1" then the program will append a version number to the custom title.

CustomLabel (optional; default: empty) - If this value is not empty the program will place this text above the uptime label.

ExitPassword (optional; default: empty) - If this value is not empty and the "enabled" attribute is "1" the program will ask for a password when a user close the program. If the "blockuserlogout" attribute is "1" the program will try to block computer reboot or user log out.

StopPassword (optional; default: empty) - If this value is not empty and the "enabled" attribute is "1" the program will ask for a password when a user click the "Stop" button.

4.4 HTTP server

This XML node contains the configuration of the HTTP server itself.

Example:

```
<HTTPServer port="80;443" ssl="0" auth="1" staticfilespath="files\" log="1"
logdetailed="0">
```

port (optional; default: 80) - One or more TCP ports. Use the semicolon (";") as a delimiter. The HTTP server will listen to all these ports.

ssl (optional; default: 0) - If "1" (true), the server will enable the SSL protocol for the selected port. If this parameter is "0" but the port number is 443, the HTTP server will always use the SSL protocol.

websocket (optional; default: 0) - If "1" (true), the HTTP server will work in the WebSocket mode. This mode allows you to get instant notifications about changes in the value of subscribed items on your web page.

auth (optional; default: 0) - If "1" (true), the HTTP server will use the "Basic" HTTP authentication for all HTTP requests. The "Users" node contains a list of Login/Password pairs.

staticfilespath (optional; default: files\) - A relative or absolute path to a folder with static files. The server will return that path in response to a GET request.

log (optional; default: 1) - If "1" (true), the HTTP server will write messages to the log file.

logdetailed - If "1" (true), the HTTP server will write detailed messages to the log file.

Users

The XML node contains a list of authorized users.

Example:

```
<User login="admin" password="admin" />
```

AllowedIPs

The XML node contains a list of allowed IP addresses. If the list is empty, the HTTP server will never check the IP addresses of the clients.

You can use the wildcard characters "?" and "*" to specify a range of IP addresses.

Example:

```
<AllowedIPs>
 <IP>192.168.1.*</IP>
</AllowedIPs>
```

192.168.1.* - Allows all IP addresses from 192.168.1.0 to 192.168.1.255

AllowedOPC

The XML node contains a list of allowed OPC servers for a user. If the list is empty, a user can connect and execute requests to all servers.

You can use the wildcard characters "?" and "*" to specify a group of similar OPC servers.

server_id (required) - it is the server ID or class ID.

host (default: empty) - it is the host name or IP address with the OPC server. If the host field is empty, the server is located locally.

Please note, the program checks access only if you defined users in the "Users" node described above.

Example:

```
<AllowedOPC>
 <!-- remove all "OPC" nodes if you want to allow access for all. Mask characters *
and ? are allowed -->
 <OPC server_id="opcserversim.Instance.*" host="*">
  <User login="*" />
 </OPC>
</AllowedOPC>
```

The example above allows access for all user to an OPC server with a name starting with "opcserversim.Instance".

4.5 SSL certificates

If you use the SSL protocol with your HTTP server, you need to prepare SSL certificates and place them in the program folder.

01ServerCert.pem - The certificate file.
01ServerKey.pem - The server key file.

The program contains self-signed certificates for the "127.0.0.1" host address.

You can use an OpenSSL distribution to prepare your own self-signed certificates.

The disadvantage of using self-signed certificates is that you have to trust them in your web browser.

You can also purchase trusted SSL certificates from an authorized reseller.

4.6 WebSocket

You may find the example for the "WebSocket" mode in the "demo_websocket.html" file, which installs in the program folder\files.

In this mode, all data packets have the JSON format. The server ignores regular HTTP requests. The server sends notifications for subscribed OPC tags automatically without additional requests.

4.7 OPC UA notes

Security and encryption

The OPC-HTTP Gateway supports the following security modes and encryptions for the OPC UA server's endpoints.

- None
- Sign
- Sign & Encrypt

- Basic128Rsa15
- Basic256
- Basic256Sha256

OPC-HTTP Gateway automatically selects the lowest security mode and encryption. For example, if the OPC UA server allows unsecured connections, our software prefers to use the "None" security mode because it requires fewer resources on the OPC server's side.

Certificate

OPC-HTTP Gateway has the following built-in certificate that you should trust in your OPC server.

```
Application Uri:urn:aggsoft.com.opc.client.application
Serial Number: 00f8e216f16b422e72
Signature algorithm: sha256RSA
Issuer: CN = aggsoft.com.opc.client@localhost;O = aggsoft.com;C = DE
Sucject: CN = aggsoft.com.opc.client@localhost;O = aggsoft.com;C = DE
Thumbprint: 2603744fb26c3345fce0f6880e451fe76b9e19c0
```

5 Service mode

The program can work in the Windows Service mode and start automatically with Windows. If the program is already running in the service mode, you cannot start other instances of it in the desktop mode.

Install/Uninstall

To install or uninstall the service, execute one of the following commands in the "cmd.exe" command prompt. If you have Windows Vista or higher, you need to run "cmd.exe" with elevated administrator privileges.

Install: opchttpgate.exe /AI
Uninstall: opchttpgate.exe /R

Start/Stop

To start or stop the service, open the "Services" control panel, find "OPC HTTP Gateway Service" there, and execute the necessary command.

More commands

To get a list of possible commands, execute the following command in the "cmd.exe" command prompt:

opchttpgate.exe /?

6 HTTP requests

6.1 Common information

HTTP mode

1. The HTTP server accepts GET and POST requests.
2. POST request type: "application/x-www-form-urlencoded".
3. If a POST request contains a query string and POST data, the HTTP server will search for the value in the "Post Data" part first, and then in the query string.
4. The HTTP server returns a JSON response with the "Content-Type" "application/json; charset=utf-8" for all valid requests.
5. The HTTP server returns the 400 (Bad Request) HTTP response code for any invalid request (for example, if some parameters or the server name are invalid).
6. The HTTP server returns the 403 (Access Denied) HTTP response code for any unauthorized request (if you have enabled this feature in the server configuration).
7. The HTTP server does not distinguish between different clients' requests. If one client subscribes to tags, another client can execute the "unsubscribe" request for the same tags.

WebSocket mode

1. The HTTP server accepts standard WebSocket messages.
2. The message format is JSON in the UTF-8 encoding.
3. The HTTP server disconnects an unauthorized client without any responses (if you have enabled this feature in the server configuration).
4. The HTTP server returns error messages in a response in the JSON format.
5. The HTTP server distinguishes different clients' requests. If one client subscribes to tags, another client will not receive notifications for these OPC tags.

6.2 Data types

The following data types can be used in the "write" request and some responses.

Data type code	Description
13	Unknown
1	Null
2	Smallint
3	Integer
4	Single
5	Double
6	Currency
7	Date

8	String
11	Boolean
16	ShortInt
17	Byte
18	Word
19	LongWord
20	Int64
21	Word64

6.3 Value quality

The Quality of a value in the OPC UA or OPC DA server is represented as a StatusCode.

High two bits define the quality type:

0x00 - Bad quality.
0x40 - Uncertain.
0xC0 - Good.

Low six bits define a sub-type.

For the Bad quality:

0x04 - Invalid or bad configuration (CONFIG ERROR).
0x08 - Not connected.
0x0C - Device failure.
0x10 - Sensor failure.
0x14 - Last known value.
0x18 - Communication error between an OPC server and a device.
0x1C - Out of service.

For the Uncertain quality:

0x44 - Last usable.
0x50 - Sensor not accurate.
0x54 - Engineering units exceeded.
0x58 - Sub normal.

For the Good quality:

0xD8 - Local override.

6.4 JSON response items

success - May contain "true" or "false"; means a response type.

processed - Contains the number of data items that have been processed by the corresponding request. If this value is greater than the size of the "data" array, it means that some items did not match the input parameters.

data - Contains an array of data items. Each data item can contain the following elements. The content of the data item depends on the request.

name - The name of an OPC tag or OPC server.

group - The group name of an OPC tag.

value - The value of an OPC tag.

data_type - The data type of the value above.

quality - The OPC value quality.

timestamp - The time stamp of the last value change (UTC+0 timezone). If an OPC server does not conform the OPC standard, it may return the timestamp in the local timezone. Our gateway returns a timestamp without any conversions.

update_count - How many times the value was updated between readings via API. If the value didn't change after the last reading, the counter would be zero. Please note that the program also counts identical values received from an OPC server.

clsid - A unique GUID of an OPC server.

desc - The description of an OPC server.

path - (OPC UA only) contains the full browse path for an OPC tag. Each node is separated by the "\u000D" ASCII character (CR).

6.5 Get Server List

This request allows you to get a list of OPC servers from the specified host.

Request page: get-servers.json

Parameters:

host (optional, case insensitive) - The host IP address. If this parameter is empty, the HTTP server will return a list of local servers.

Example for the HTTP server:

http://127.0.0.1/get-servers.json?host=192.168.1.180
http://127.0.0.1/get-servers.json

Example for the WebSocket server:

```
{"command":"get-servers", "host":"192.168.1.180"}
{"command":"get-servers"}
```

Response example:

```
{"command":"get-servers", "success": true, "processed": 1, "data":
[{"name": "opcserversim.Instance.1", "desc": "Demo server", "clsid": "{0a9db5fd-
4ca5-4611-84fb-3f5b75692fa9}", }]}
```

OPC UA: The program connects to a discovery server to get a list of servers. Therefore, you should install Local Discovery Server by OPC Foundation on a local or remote computer, and your OPC UA server should register itself on that server.

6.6 Get Tag List

This request allows you to get a list of tags from the specified OPC server.

Request page: get-tag-tree.json

Parameters:

server (required, case sensitive) - The OPC server name.

host (optional, case insensitive) - The host IP address of the OPC server. The OPC UA gateway ignores this parameter because the server name already contains the full path.

Example for the HTTP server:

http://127.0.0.1/get-tag-tree.json?server=opcserversim.Instance.1

Example for the WebSocket server:

```
{"command":"get-tag-tree", "server":"opcserversim.Instance.1"}
```

Example for the WebSocket server and OPC UA:

```
{"command":"get-tag-tree",
"server":"opc.tcp://192.168.1.3:16664/OpenOpcUaCoreServer"}
```

Response example for OPC DA:

```
{"command":"get-tag-tree", "success": true, "data":
[{"name": "Channel1.Device1.Tag1", "data_type": 8},
{"name": "Channel1.Device1.Tag2", "data_type": 3},
{"name": "Channel1.Device1.Tag3", "data_type": 3},
{"name": "Channel2.Device1.Tag1", "data_type": 3}]}
```

Response example OPC UA:

```
{"command":"","success":true,"data":
[{"name":"NS4|string|
ByteString","display_name":"ByteString","data_type":0,"path":"NonUaNodeComplianceTe
st"},
{"name":"NS4|string|
SByte","display_name":"SByte","data_type":0,"path":"NonUaNodeComplianceTest"},
{"name":"NS4|string|
Time","display_name":"Time","data_type":0,"path":"NonUaNodeComplianceTest"},
{"name":"NS4|string|
String","display_name":"String","data_type":0,"path":"NonUaNodeComplianceTest"}]}
```

OPC UA tag name format

OPC tag name corresponds to the node id in the OPC server's address space. It contains three parts.

1. Namespace ID (for example, NS4).
2. Node id type (number, string, octet, guid).
3. The identifier (for example, `ByteString`).

The "**path**" attribute contains a browse path for the corresponding tag where each level is separated by the "\" characters.

6.7 Subscribe

This request allows you to add a tag to a subscription list. The server will poll the tag value periodically (every 100 ms for local OPC servers, or every 1000 ms for remote OPC servers) and store the OPC tag value internally. Subsequent "read" requests will get the value from the internal buffer, so they will work much faster.

Request page: subscribe.json

Parameters:

server (required, case sensitive) - The OPC server name.

host (optional, case insensitive) - The host IP address of the OPC server.

group (required, case sensitive) - The group name. This name is not related to the OPC group’s name on the OPC server. You can use this group name as a shortened version for the "read" request. For example, you can read a whole group.

tag (required, case sensitive) - The OPC server tag name (names). You can get these names using the "get-tag-list" request.

OPC DA:
You can specify multiple tag names by using the "|" (pipe) or "\u000D" character as a delimiter.

OPC UA:
You can specify multiple tag names by using the "\u0009" (tab) or "\u000D" (Cr) character as a delimiter.

Example for the HTTP server:

http://127.0.0.1/subscribe.json?server=opcserversim.Instance.1&group=MyGroup1&tag=Channel1.Device1.Tag1|Channel1.Device1.Tag2

Example for the WebSocket server:

```
{"command":"subscribe", "server": "opcserversim.Instance.1", "group": "MyGroup1",
"tag": "Channel1.Device1.Tag1|Channel1.Device1.Tag2"}
{"command":"subscribe", "server": "opcserversim.Instance.1", "group": "MyGroup1",
"tag": "Channel1.Device1.Tag1\u000DChannel1.Device1.Tag2"}
```

Response example:

```
{"command":"subscribe", "success": true, "processed": 1}
```

Or

```
{"command":"subscribe", "success": false, "processed": 0}
```

6.8 Unsubscribe

This request is an opposite of the "subscribe" request and allows you to remove tags from the subscription list. If the HTTP server removes all connected tags from the subscription list, the HTTP server will close the connection with the OPC server.

Request page: unsubscribe.json

Parameters:

server (required, case sensitive) - The OPC server name.

host (optional, case insensitive) - The host IP address of the OPC server.

group (optional, case sensitive) - The group name from the "subscribe" request. You can omit the "tag" value in this request. In this case, all tags for this group will be removed from the subscription list. If the the "group" is equal to "*", all tags in all groups will be removed.

tag (optional, case sensitive) - The list of tags (see the "subscribe" request). If you specify this parameter, the group parameter will not be used.

Example for the HTTP server:

http://127.0.0.1/unsubscribe.json?server=opcserversim.Instance.1&tag=Channel1.Device1.Tag1|Channel1.Device1.Tag2
http://127.0.0.1/unsubscribe.json?server=opcserversim.Instance.1&group=MyGroup
http://127.0.0.1/unsubscribe.json?server=opcserversim.Instance.1&group=*

Example for the WebSocket server:

```
{"command":"unsubscribe", "server": "opcserversim.Instance.1", "tag":
"Channel1.Device1.Tag1| Channel1.Device1.Tag2"}
{"command":"unsubscribe", "server": "opcserversim.Instance.1", "group": "MyGroup"}
{"command":"unsubscribe", "server": "opcserversim.Instance.1", "group": "*"}
```

Response example:

```
{"command":"unsubscribe", "success": true, "processed": 1}
```

Or

```
{"command":"unsubscribe", "success": false, "processed": 0}
```

6.9 List Subscribed

This request allows you to get a list of tags from the internal subscription list.

Request page: list-subscribed.json

Parameters:

server (required, case sensitive) - The OPC server name.

host (optional, case insensitive) - The host IP address of the OPC server.

group (required, case sensitive) - The group name from the "subscribe" request. If the "group" is equal to "*",all tags in all groups will be processed.

Example for the HTTP server:

http://127.0.0.1/list-subscribed.json?server=opcserversim.Instance.1&group=*

Example for the WebSocket server:

```
{"command":"list-subscribed", "server": "opcserversim.Instance.1", "group": "*"}
```

Response example:

```
{"command":"list-subscribed", "success": true, "data":
[{"name": "Channel1.Device1.Tag1", "group": "MyGroup"},
{"name": "Channel1.Device1.Tag2", "group": "MyGroup"},
{"name": "Channel1.Device1.Tag3", "group": "MyGroup"}]}
```

6.10 Read

This request allows you to read an OPC tag value from the specified OPC server. If the specified OPC tag exists in the subscription list, the HTTP server will return the cached value.

If you try to read a single tag that does not exist in the subscription list, the server will connect to the server and read the value using the synchronous method (SyncRead). It may take a while and depends on the OPC server implementation.

Request page: read.json

Parameters:

server (required, case sensitive) - The OPC server name.

host (optional, case insensitive) - The host IP address of the OPC server.

group (optional, case sensitive) - The group name from the "subscribe" request. You can omit the "tag" value in this request. In that case, the server will return values for all tags in the specified group. If the the "group" is equal to "*",the server will return values for all tags in all groups.

tag (optional, case sensitive) - The list of tags (see the "subscribe" request). If you specify this parameter, the server will ignore the "group" parameter.

Example for the HTTP server:

http://127.0.0.1/read.json?server=opcserversim.Instance.1&tag=Channel1.Device1.Tag1

Example for the WebSocket server:

```
{"command":"read", "server": "opcserversim.Instance.1", "tag":
"Channel1.Device1.Tag1"}
```

Response example:

```
{"command":"read", "success": true, "processed": 1,
"data": [{"name": "Channel1.Device1.Tag1", "group": "MyGroup", "value"
: "123", "quality": "0xC0", "timestamp": "2016-10-27 06:23:24", "data_type": 8}]}
```

Response example of the WebSocket server for a new value:

```
{"command":"read", "async": true, "success": true, "processed": 1,
"data": [{"name": "Channel1.Device1.Tag1", "group": "MyGroup", "value"
: "123", "quality": "0xC0", "timestamp": "2016-10-27 06:23:24", "data_type": 8}]}
```

Response

data - Contains an array of data items. Each data item can contain the following elements. The content of the data item depends on the request.

name - The name of an OPC tag or OPC server.
group - The group name of an OPC tag.

value - The value of an OPC tag.
data_type - The data type of the value above.
quality - The OPC value quality.
timestamp - The time stamp of the last value change in an OPC server.

6.11 Write

This request allows you to write a new value to a single OPC tag on the specified OPC server. If you try to write to a tag that does not exist in the subscription list, the program will connect to the server and try to write the value.

Request page: write.json

Parameters:

server (required, case sensitive) - The OPC server name.

host (optional, case insensitive) - The host IP address of the OPC server.

group (optional, case sensitive) - The group name from the "subscribe" request. The server searches for a subscribed tag using this group name.

tag (required, case sensitive) - An OPC tag name.

value (required, case sensitive) - A new value.

data_type (optional, number only) - The data type of the specified value. You can omit this parameter for subscribed tags. In that case, the server will use the data type of the cached value.

Example for the HTTP server:

http://127.0.0.1/write.json?
server=opcserversim.Instance.1&group=MyGroup&tag=Channel1.Device1.Tag1&value=321&data_typ
e=8

Example for the WebSocket server:

```
{"command":"write", "server": "opcserversim.Instance.1", "group": "MyGroup", "tag":
"Channel1.Device1.Tag1", "value": "321", "data_type": 8}
```

Response example:

```
{"command":"write", "success": true, "processed": 1}
```

6.12 Demo

We have prepared a demo page that uses jQuery and illustrates all features of the program. We have placed the "demo.html" page in the "files" folder with all static files.

Just start the program and open the following URL in your web browser:

HTTP server: http://127.0.0.1:8080/demo.html
HTTPS server: https://127.0.0.1:8443/demo.html (by default, the server uses self-signed SSL certificates that you should trust in your browser)
HTTP UA server: http://127.0.0.1:8080/demo_http_ua.html

WebSocket server: http://127.0.0.1:8080/demo_websocket.html

Please note, the links above contain a port number. It can be changed in the OPC-HTTP Gateway configuration. Therefore, the link above may not work.