

wsLogger implementation infos

The task „wsLogger“ provides entries of a logger module as JSON structure, internally using the „ReadEvLog“ library plus some BR libraries like AsHttp.

The maximum of the last 1000 entries of a logger module can be provided.

The datamodule „wsLData“ additionally contains a simple frontend for the webservice provided by „wsLogger“. Because of technical reasons, the web frontend code is byte-coded and cannot be edited directly.

The provisioning of the frontend code by a datamodule has the advantage, that no extra files on the user partition for the AR integrated webserver are needed, which makes it simple to deploy everything by Automation Studio online download.

If you want to see how the frontend code works, just:

- * open the frontend in your browser
- * store the frontend code to a file (the code has a jQuery instance embedded) and inspect it
- * build your own frontend code and use it with the AR embedded webserver by transferring your code to the user partition web root directory
- * from „wsLogger“, just use the webservice with parameters, instead of using it with the built-in frontend.

Without any changes in „wsLogger“, the frontend can be accessed by:

http://<IP Adress of your PLC>/getLogger.cgi

Example:

The screenshot shows the wsLogger web frontend in a browser window. The address bar displays `http://192.168.134/getLogger.cgi`. The page title is "wsLogger". Below the title, there is a section for "line format" with fields for RecordID, OriginRecordID, DateTime.Nanoseconds, Severity, EventID, ObjectID, and AddData. Below this, there are checkboxes for "Search here:" and "Search for:". There are also buttons for "(Re)Load logger names from PLC" and "(Re)Load data from PLC". A search bar with "search for..." and a "Start Search" button is present. A hint at the bottom states: "max. 1000 latest entries are read from the webservice inside the PLC" and "hint: the search result without 'don't hide...' shows also all parent entries of a found string in a line - this is not a bug :-)".

All loggers existing should be available via dropdown, and by choosing one you can load and display the data (if logger entries contain dependencies, the hirarchy is also displayed)

This screenshot shows the same wsLogger web frontend, but with a dropdown menu open for the "logger modules on PLC" field. The dropdown list contains the following items: \$accsec, \$accsec, \$arlogconn, \$arlogsys, \$arlogusr, \$fieldbus, \$firewall, \$motion, \$redund, \$safety, \$stextsys, \$unitsys, \$versinfo, and myLogger. The "name of the logger module" field is currently set to \$accsec. The rest of the interface, including the search filters and buttons, remains the same as in the previous screenshot.

The screenshot shows the wsLogger web interface. The search criteria are set to 'myLogger' and '(Re)Load data from PLC'. The search results are displayed as a list of log entries. The entry for 'loop run 878' is highlighted in yellow. The entry text is: '878 878 DT#2025-11-30-11:13:39.441000000 2 -1609956502 TestDataGenerator this is my custom text - loop run 878'.

Also, some searching inside the data is possible:

The screenshot shows the wsLogger web interface. The search criteria are set to 'myLogger' and '(Re)Load data from PLC'. The search results are displayed as a list of log entries. The entry for 'this is my custom text' is highlighted in yellow. The entry text is: '878 878 DT#2025-11-30-11:13:39.441000000 2 -1609956502 TestDataGenerator this is my custom text - loop run 878'.

If you want to use just the webservice data without the frontend, here are the parameters.

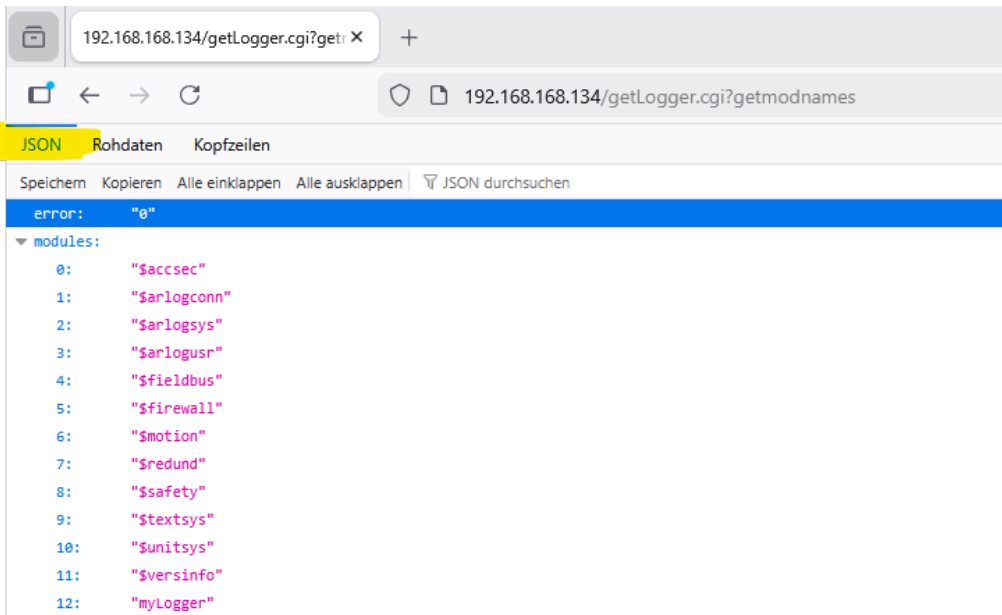
request:

http://<IP Address of your PLC>/getLogger.cgi?getmodnames

response:

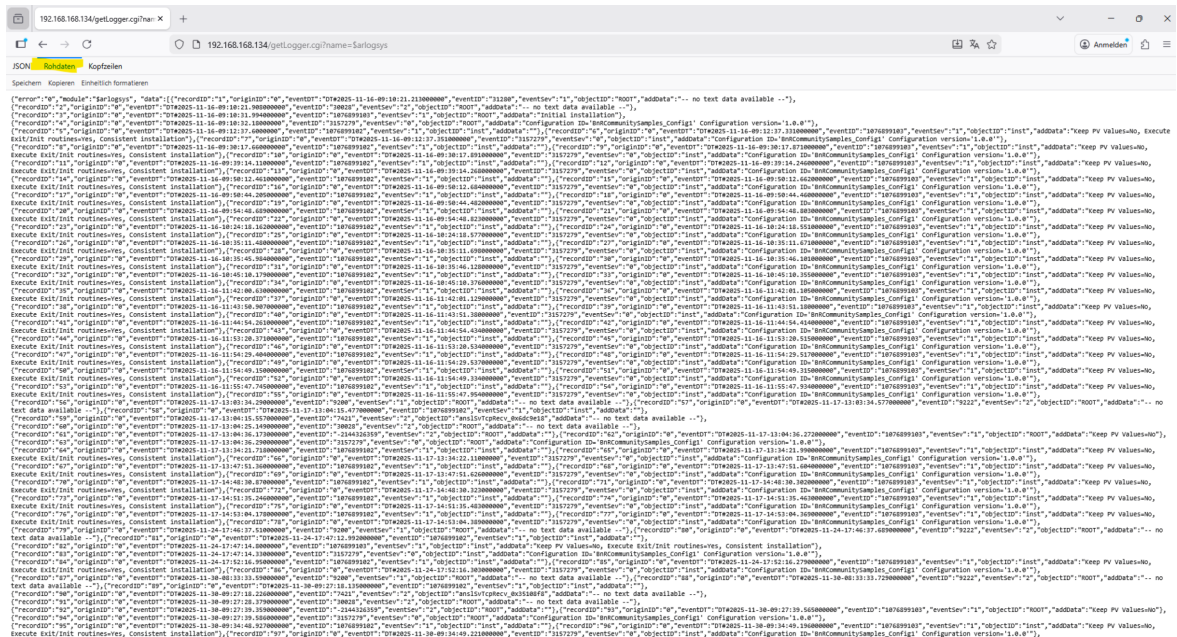
the logger modules available on the PLC (JSON formatted)

The screenshot shows a web browser displaying the JSON response from the wsLogger webservice. The response is formatted as JSON and includes the following modules: "sacsec", "sarlogconn", "sarlogsys", "sarlogusr", "sfieldbus", "sfirewall", "smotion", "sredund", "ssafety", "stextsys", "sunitsys", "sversinfo", and "myLogger".



request:
http://<IP Adress of your PLC>/getLogger.cgi?name=<Name of the logger module>

response:
the logger module content (JSON formatted)



192.168.168.134/getLogger.cgi?na: X +

192.168.168.134/getLogger.cgi?name=\$arlogsys

JSON Rohdaten Kopfzeilen

Speichern Kopieren Alle einklappen Alle ausklappen JSON durchsuchen

```
{
  "error": "0",
  "module": "$arlogsys",
  "data": [
    {
      "recordID": "1",
      "originID": "0",
      "eventDT": "DT#2025-11-16-09:10:21.213000000",
      "eventID": "31280",
      "eventSev": "1",
      "objectID": "ROOT",
      "addData": "-- no text data available --"
    },
    {
      "recordID": "2",
      "originID": "0",
      "eventDT": "DT#2025-11-16-09:10:21.908000000",
      "eventID": "30028",
      "eventSev": "2",
      "objectID": "ROOT",
      "addData": "-- no text data available --"
    },
    {
      "recordID": "3",
      "originID": "0",
      "eventDT": "DT#2025-11-16-09:10:31.994000000",
      "eventID": "1076899103",
      "eventSev": "1",
      "objectID": "ROOT",
      "addData": "Initial installation"
    },
    {
      "recordID": "4",
      "originID": "0",
      "eventDT": "DT#2025-11-16-09:10:32.180000000",
      "eventID": "3157279",
      "eventSev": "0",
      "objectID": "ROOT",
      "addData": "Configuration ID='8nrCommunitySamples_Config1' Configuration version='1.0.0'"
    },
    {
      "recordID": "5",
      "originID": "0",
      "eventDT": "DT#2025-11-16-09:12:37.600000000",
      "eventID": "1076899102",
      "eventSev": "1",
      "objectID": "Inst",
      "addData": ""
    },
    {
      "recordID": "6",
      "originID": "0",
      "eventDT": "DT#2025-11-16-09:12:37.600000000",
      "eventID": "3157279",
      "eventSev": "0",
      "objectID": "ROOT",
      "addData": "Configuration ID='8nrCommunitySamples_Config1' Configuration version='1.0.0'"
    }
  ]
}
```