S7-Web-LINK

S7-Web-LINK provides data from an SPS using a Json Rest API

Getting Started

Edit configuration file

S7-Web-LINK.config

```
{
    "WebServerSettings": {
        "Port": 8080,
        "UseHTTPS": true,  // Wheter SLL (HTTPS) should be used or not
        "Certificate": "S7-Web-LINK.pem"  // Path to certificate file
    },
    "Connections": {
        "SPS-1": {  // Predefined connection see chapter "Request -> Connection"
            "IPAddr": "127.0.0.1",
            "PlcType": "s7_300_400",
            "Channel": "op",
            "Rack": ,
            "Slot": 2,
            "RxTimeout": 5000,
            "TxTimeout": 5000,
            "ConTimeout": 5000
        }
    },
    "TokenList": { // SPS with the IP "127.0.0.1" could be accessed using "Auth":
"password"
        "127.0.0.1": "password",
        "192.168.2.123": "secret",
        "192.168.0.80": "123456"
    }
}
```

Generate SSL Certificate (optional)

Sample Certificate

S7-Web-LINK.pem

```
-----BEGIN CERTIFICATE-----
MIICWDCCAcGgAwIBAgIJAIfbmjDRjQcwMA0GCSqGSIb3DQEBCwUAMEUxCzAJBgNV
BAYTAkRFMRMwEQYDVQQIDApTb21lLVN0YXRlMSEwHwYDVQQKDBhJbnRlcm5ldCBX
aWRnaXRzIFB0eSBMdGQwHhcNMTYwMjIyMTIyNjMxWhcNMTcwMjIxMTIyNjMxWjBF
MQswCQYDVQQGEwJERTETMBEGA1UECAwKU29tZS1TdGF0ZTEhMB8GA1UECgwYSW50
ZXJuZXQgV2lkZ2l0cyBQdHkgTHRkMIGfMA0GCSqGSIb3DQEBAQUAA4GNADCBiQKB
gQC/u0VOmGadfg7WsGSDi8SxBl+KCALRw24DHFotnsiSBAp1st+TNd8wzrpjAsLb
1f1ZnmN2/PthSGzorhMGJC4iMXcDk1TmDLgDuoTovq8dJ/vqXcHaEr+T59vJAn+s
NR2tR7PlARykWTCO0qizcigW1ICIyEmYkuQokeZsHAaqWQIDAQABo1AwTjAdBgNV
HQ4EFgQUVVSEEuEv+mAbb/C4wEZYjxbucbMwHwYDVR0jBBgwFoAUVVSEEuEv+mAb
b/C4wEZYjxbucbMwDAYDVR0TBAUwAwEB/zANBgkqhkiG9w0BAQsFAAOBgQCw+IaN
GMYDwonsMstcXs5DC9lqvVGBnBPF3biF76q/MsKy+zytFmFsPsqlV/OMRLstTC+0
WDIB6WaoHpBrzJopxVOZQQczbdqWfUcpI++6Q8o6peBtzyAqa5IAygfia3yMOI7R
laxVSfbEqEzPVow8QcaiewLWqEFGBOlsIVDv8w==
-----END CERTIFICATE-----
-----BEGIN RSA PRIVATE KEY-----
MIICXAIBAAKBgQC/u0VOmGadfg7WsGSDi8SxBl+KCALRw24DHFotnsiSBAp1st+T
Nd8wzrpjAsLb1f1ZnmN2/PthSGzorhMGJC4iMXcDk1TmDLgDuoTovq8dJ/vqXcHa
Er+T59vJAn+sNR2tR7PlARykWTCO0qizcigW1ICIyEmYkuQokeZsHAaqWQIDAQAB
AoGAQvK9TKhijHPL8qM9NcHEOJwlGCmb8mrvKx7nTi63kmTe0iJXdyvEd2J4KsJ4
EBM0l+p6iL3leR61CLpf4jEX+jUNJgFgduwLpXMeq/jaNAxNRRsiDEnmFqIMZGYX
R+9/e2dTlJFAe/VAOuMxylLUVqWKQpI/3eaQRo0GkRUMYAECQQD1UCvntG0D7e1u
ksh8QUwY4A28FvLbiCxYK0jpw15iKDz/DlwhrfZB8I5EgVQMWXJWcGUWSzbEvqRr
4P+6SakBAkEAyBWJJ7qDL7DgqIbgaQWLicqhKF3mO1nxj5DtEtUdu/cP0liGiV8M
tlMyf/YWuI9l3qmfMUXXjaQjkavFTJnpWQJAP/wnV4piHPJESeETVgWaGarnKjY6
JjTAjEbN/9srlSK1tjlCoq5DWzOpiAjLqWTzQ8SROV1o7axkKpdHXIm2AQJAatCs
bwwwOsaXuQCATzDJpJ8LWAyA+9BxmC5LkhE6FX248ZeXA0E9/Rv/SrbqvE65mJw/
Q0PA5nnpDzx/UPydyQJBAMH3ZdrdxwF2rvdXtjNYq1NkG2N783Uok6WQ55mDiBhn
sBd9qHiTnGsjZHFJasb01y5w87ZxE6aDdvMO0XCdKOY=
-----END RSA PRIVATE KEY-----
```

Windows

Download SSL Manager

Linux

Install OpenSSL for Debian or Ubuntu based distribute:

```
sudo apt-get install openssl
```

Install on other linux distribution

- GenerateKey.sh

```
openssl genrsa -out server.key 1024
openssl req -days 365 -out server.pem -new -x509 -key server.key
cat server.pem >> S7-Web-LINK.pem
cat server.key > S7-Web-LINK.pem
```

Sample

This will read a value from our SPS using JavaScript

PHP Example

With cURL

```php
<?php
// The data to send to the API
$postData = array(
    "Auth" => "password",
    "Connection" => array(
        "IPAddr" => "127.0.0.1"
    ),
    "Read" => array(
        array(
            "Addr" => "DB1000.DBB 304",
            "Type" => "string",
            "Length" => 10
        ),
        array(
            "Addr" => "DB1000.DBX 322.0",
            "Type" => "byte"
        )
    )
);

// Setup cURL
$ch = curl_init("https://127.0.0.1");
curl_setopt_array($ch, array(
    CURLOPT_POST => TRUE,
    CURLOPT_RETURNTRANSFER => TRUE,
    CURLOPT_HTTPHEADER => array(
        'Content-Type: application/json'
    ),
    CURLOPT_POSTFIELDS => json_encode($postData)
));

// Send the request
$response = curl_exec($ch);

// Check for errors
if($response === FALSE){
    die(curl_error($ch));
}

// Decode the response
$responseData = json_decode($response, TRUE);

// Print the date from the response
var_dump($responseData);
?>
```

Without cURL

```php
<?php
// The data to send to the API
$postData = ...

// Create the context for the request
$context = stream_context_create(array(
    "http" => array(
        // http://www.php.net/manual/en/context.http.php
        "method" => "POST",
        "header" => "Content-Type: application/json\r\n",
        "content" => json_encode($postData)
    )
));

// Send the request
$response = file_get_contents("https://127.0.0.1", FALSE, $context);

// Check for errors
if($response === FALSE){
    $error = error_get_last();
    var_dump($error);
    die();
}

echo $response . "<br />";

// Decode the response
$responseData = json_decode($response, TRUE);

// Print the date from the response
var_dump($responseData);
?>
```

Request

Connection

```
{
    "Auth": "password",             // Password or Token
    "Connection": "SPS-1",          // Name predefined in the configuration file
    "Read": [ ... ],                // optional
    "Write": [ ... ]                // optional
}
```

```
{
    "Auth": "password",              // Password or Token
    "Connection": {
        "IPAddr": "127.0.0.1",
        "PlcType": "s7_300_400",     // optional, could be set to "s7_300_400" (default),
"s7_1200", "s7_1500", "s7_200", "s7_logo"
        "Channel": "op"              // optional, could be set to "op" or "pg"
        "Rack": ,                   // optional
        "Slot": 2,                   // optional
        "RxTimeout": 5000,           // optional, default 5000
        "TxTimeout": 5000,           // optional, default 5000
        "ConTimeout": 5000           // optional, default 5000
    },
    "Read": [ ... ],                 // optional
    "Write": [ ... ]                 // optional
}
```

Read

```
{
    "Auth": "password",
    "Connection": { ... },
    "Read": [{
        "Addr": "DB1000.DBB 304",
        "Type": "string",
        "Length": 10
    },
    {
        "Addr": "DB1000.DBX 322.0",
        "Type": "byte"
    }]
}
```

Write

```
{
    "Auth": "password",
    "Connection": { ... },
    "Write": [{
        "Addr": "DB1000.DBB 304",
        "Type": "string",
        "Value": "HelloWorld"
    },
    {
        "Addr": "DB1000.DBD 300",
        "Type": "double",
        "Value": 13.7
    }]
}
```

Read / Write Types

Name	Bit
byte	8 (unsigned)
word	32 (unsigned)
word16	16 (unsigned)
word32	32 (unsigned)
word64	64 (unsigned)
int	32 (signed)
int16	16 (signed)
int32	32 (signed)
int64	64 (signed)
float	32 (signed)
double	64 (signed)
string	

Response

Connection

```
{
    "Connection": {
        "PlcType": ...,             // echo (same as request)
        "Channel": ...              // echo (same as request)
        "IPAddr": ...,              // echo (same as request)
        "Rack": ...,                // echo (same as request)
        "Slot": ...,                // echo (same as request)
        "RxTimeout": ...,           // echo (same as request)
        "TxTimeout": ...,           // echo (same as request)
        "ConTimeout": ...,          // echo (same as request)
        "Result": {
            "Value": -2,            // default 0
            "Message": "The maximum number (256) of the possible connections is reached!" //
default "OK"
        }
    },
    "Read": [ ... ],                // optional
    "Write": [ ... ]                // optional
}
```

Read

```
{
    "Connection": { ... },
    "Read": [{
        "Addr": "DB1000.DBB 304",    // echo (same as request)
        "Type": "string",            // echo (same as request)
        "Length": 10,                // echo (same as request)
        "Value": "HelloWorld",
        "Result": {
            "Value": ,              // default 0
            "Message": "OK"          // default "OK"
        }
    },
    {
        "Addr": "DB1000.DBX 322.0",  // echo (same as request)
        "Type": "byte",              // echo (same as request)
        "Value": null,               // if read failed value will be null
        "Result": {
            "Value": -1,             // default 0
            "Message": "Desired data type is not allowed or is not supported." // default
"OK"
        }
    }]
}
```

Write

```
{
    "Connection": { ... },
    "Write": [{
        "Addr": "DB1000.DBB 304",    // echo (same as request)
        "Type": "string",            // echo (same as request)
        "Value": "HelloWorld",       // echo (same as request)
        "Result": {
            "Value": ,              // default 0
            "Message": "OK"          // default "OK"
        }
    },
    {
        "Addr": "DB1000.DBD 300",    // echo (same as request)
        "Type": "double",            // echo (same as request)
        "Value": 13.7,               // echo (same as request)
        "Result": {
            "Value": ,              // default 0
            "Message": "OK"          // default "OK"
        }
    }]
}
```

Error Messages

Value	Message	Description (Reaction)
0	OK	**Default** Everything okay

Value	Message	Description (Reaction)
-1	Timeout, PLC not reachable.	Start further read/write operations. The driver automatically repairs the connection. May you hav to increase the timeout values (receive/connect timeout).
-2	The maximum number (256) of the possible connections is reached.	Close unused connections
-5	General error.	Check whether network is properly installed in the PC: TCP / IP enabled? Winsocket installed?
-6	Target CPU not found.	Rack or number of slot invalid.No connection to these slot available. Check configuration in CP and PLC
-7	Socket error	`"Result": { "Value": -7, "Message": "Socket error", "SocketError": { "Value": 10013, "Message": "Permission denied." } }` Win Socket errorlist
-8	Memory error	Requested memory in the PC is not available.
-9	Overrange	e.g.. timer > 9990000 ms
-10	Desired data type is not allowed or is not supported.	Check that the code for data type is valid.
-11	The specified PC data type do not correspond to the specified PLC data area.	e.g : This can occur if you want to access counter and the PC is specified as data type BYTE. Solution: Change the data type in the PC.
-31	PC and S7 data type are in invalid relation e.g. PC = BYTE = Word PLC	Adjust PC – data area.
-32	S7 returns incorrect number of data for the specified data type.	
-33	Consecutively access to multiple bits not available.	
-34	The requested request block could not be assigned.	
-1002	No resource available.	Maximum of the available connections is reached.
2	Block or data area does not exist, for example access to DB, which is not available, or too small.	Verify that the desired data area exists in the PLC.

Table of Contents