

The background image shows a man in a light blue shirt from the side, looking at a tablet. He is in a factory or industrial setting with various machines and equipment visible in the background. Overlaid on the image are several digital graphics: a Siemens logo in the top right, a '24/7' icon with a circular arrow, a 'NEWS' icon with a person silhouette, a 'Home' icon, and a large 'Industry Online Support' text. There are also binary code (0s and 1s) and network-like icons scattered throughout the digital overlay.

# SIEMENS

## Library for Data Streams (LStream)

TIA Portal, JSON / XML Deserializer and JSON / XML Serializer

<https://support.industry.siemens.com/cs/ww/en/view/109781165>

Siemens  
Industry  
Online  
Support



# Legal information

## Use of application examples

Application examples illustrate the solution of automation tasks through an interaction of several components in the form of text, graphics and/or software modules. The application examples are a free service by Siemens AG and/or a subsidiary of Siemens AG ("Siemens"). They are non-binding and make no claim to completeness or functionality regarding configuration and equipment. The application examples merely offer help with typical tasks; they do not constitute customer-specific solutions. You yourself are responsible for the proper and safe operation of the products in accordance with applicable regulations and must also check the function of the respective application example and customize it for your system.

Siemens grants you the non-exclusive, non-sublicensable and non-transferable right to have the application examples used by technically trained personnel. Any change to the application examples is your responsibility. Sharing the application examples with third parties or copying the application examples or excerpts thereof is permitted only in combination with your own products. The application examples are not required to undergo the customary tests and quality inspections of a chargeable product; they may have functional and performance defects as well as errors. It is your responsibility to use them in such a manner that any malfunctions that may occur do not result in property damage or injury to persons.

## Disclaimer of liability

Siemens shall not assume any liability, for any legal reason whatsoever, including, without limitation, liability for the usability, availability, completeness and freedom from defects of the application examples as well as for related information, configuration and performance data and any damage caused thereby. This shall not apply in cases of mandatory liability, for example under the German Product Liability Act, or in cases of intent, gross negligence, or culpable loss of life, bodily injury or damage to health, non-compliance with a guarantee, fraudulent non-disclosure of a defect, or culpable breach of material contractual obligations. Claims for damages arising from a breach of material contractual obligations shall however be limited to the foreseeable damage typical of the type of agreement, unless liability arises from intent or gross negligence or is based on loss of life, bodily injury or damage to health. The foregoing provisions do not imply any change in the burden of proof to your detriment. You shall indemnify Siemens against existing or future claims of third parties in this connection except where Siemens is mandatorily liable.

By using the application examples you acknowledge that Siemens cannot be held liable for any damage beyond the liability provisions described.

## Other information

Siemens reserves the right to make changes to the application examples at any time without notice. In case of discrepancies between the suggestions in the application examples and other Siemens publications such as catalogs, the content of the other documentation shall have precedence.

The Siemens terms of use (<https://support.industry.siemens.com>) shall also apply.

## Security information

Siemens provides products and solutions with industrial security functions that support the secure operation of plants, systems, machines and networks.

In order to protect plants, systems, machines and networks against cyber threats, it is necessary to implement – and continuously maintain – a holistic, state-of-the-art industrial security concept. Siemens' products and solutions constitute one element of such a concept.

Customers are responsible for preventing unauthorized access to their plants, systems, machines and networks. Such systems, machines and components should only be connected to an enterprise network or the internet if and to the extent such a connection is necessary and only when appropriate security measures (e.g. firewalls and/or network segmentation) are in place.

For additional information on industrial security measures that may be implemented, please visit <https://www.siemens.com/industrialsecurity>.

Siemens' products and solutions undergo continuous development to make them more secure. Siemens strongly recommends that product updates are applied as soon as they are available and that the latest product versions are used. Use of product versions that are no longer supported, and failure to apply the latest updates may increase customer's exposure to cyber threats.

To stay informed about product updates, subscribe to the Siemens Industrial Security RSS Feed under <https://www.siemens.com/cert>.

# Table of contents

<b>Legal information .....</b>	<b>2</b>
<b>1 Introduction .....</b>	<b>4</b>
1.1 Overview .....	4
1.2 Functionality .....	5
1.3 Restrictions and Requirements .....	5
1.4 Components used .....	6
<b>2 Engineering .....</b>	<b>7</b>
2.1 Components of the Library.....	7
2.2 Interface description .....	8
2.2.1 LStream_JsonDeserializer.....	8
2.2.2 LStream_JsonSerializer.....	10
2.2.3 LStream_XmlDeserializer .....	12
2.2.4 LStream_XmlSerializer .....	14
2.2.5 LStream_FindStringInByteCharArrayAdv .....	16
2.2.6 LStream_typeElement .....	16
2.3 Integration into the user project.....	17
2.4 Using & understanding the demo project.....	20
<b>3 Additional information .....</b>	<b>21</b>
3.1 The JavaScript Object Notation (JSON) data exchange format in accordance with RFC 7159.....	21
3.2 JSON Representation in a Tree .....	21
3.3 Extensible Markup Language (XML) .....	22
3.4 Library in the TIA Portal .....	22
<b>4 Appendix .....</b>	<b>23</b>
4.1 Service and support.....	23
4.2 Industry Mall.....	24
4.3 Links and literature .....	24
4.4 Change documentation.....	24

# 1 Introduction

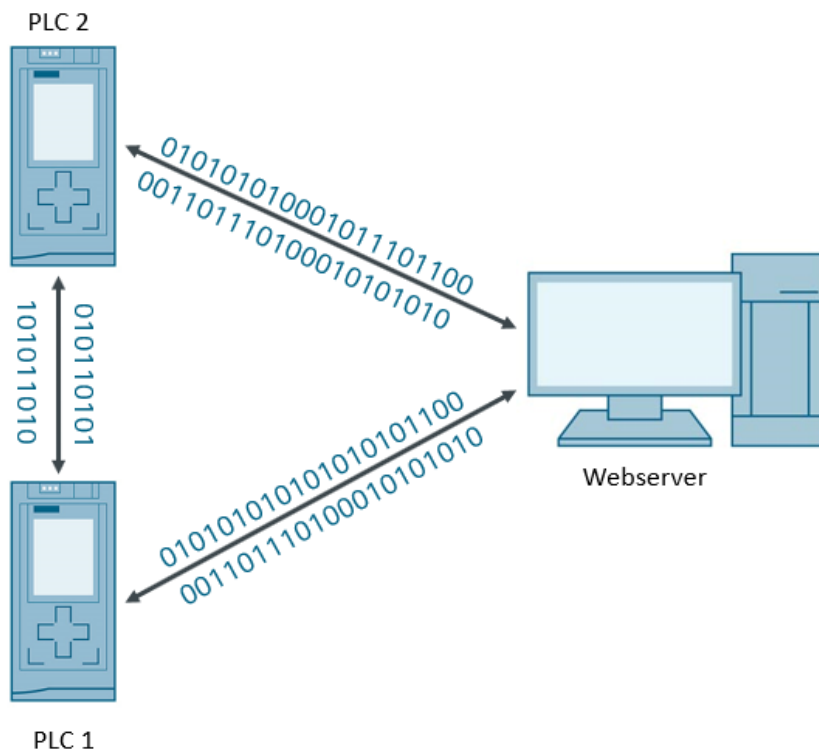
## 1.1 Overview

In computer science, a stream is a sequence of data elements that are made available over time. Examples of these data element sequences include data formats such as JSON or XML. These are increasingly being used for the exchange of data from or to a web server.

Due to the increasing use of networking within plants and the advancement of the Internet of Things (IoT), the exchange of data using JSON or XML format plays an increasingly important role in automation technology.

The LStream library makes it possible to convert the JSON and XML data formats into a format that is more compatible with a SIMATIC S7 Controller for further processing. It can also be used to create an XML format from abstract data for further processing.

Figure 1-1: Overview





## 1.2 Functionality

The LStream library provides function blocks that can be used to deserialize JSON and XML data streams for the user program and to serialize them again from the user program.

## 1.3 Restrictions and Requirements

Because of system behavior of the SIMATIC PLC and because datatypes of strings / characters have been chosen to be String / Byte (because of memory optimization) it is only possible to serialize/ deserialize streams with ASCII encoding. Any other encoding will lead to undefined behavior and might cause issues while serializing/ deserializing.

Additionally, the length of strings (e.g. values of XML / JSON elements/ attributes) is restricted to a maximum of 256 characters. There is no workaround for these restrictions; if the functionality is required the user must adjust the library code.

### Requirements JSON

The incoming JSON stream must be formatted in a compressed form:

- Line breaks are not allowed
- No spaces or whitespaces outside of values

Code example 1-1 open JSON format

```
{
  "glossary":
  {
    "title": "example glossary",
    "GlossDiv":
    [
      "XML",
      "S"
    ]
  }
}
```

Code example 1-2 compressed JSON format

```
{"glossary":{"title":"example glossary","GlossDiv":["XML","S"]}}
```

This format has been chosen to optimize the use of memory and performance for serialization and deserialization.

Serialization will always output a compressed format. When the input is not compressed the behavior of the function is undefined and might lead to issues while deserializing.

## 1.4 Components used

This library was created with these hardware and software components:

Table 1-3

Components	Quantity	Article number	Note
SIMATIC CPU 1517TF-3 PN/DP	1	6ES7517-3UP00-0AB0	FW 2.8
SIMATIC STEP 7 Professional V18 Update 1	1	6ES7822-1AA08-0YA5	-

This library consists of the following components:

Table 1-4

Components	File name	Note
Documentation	109781165_LStream_DOC_v1_6_de.pdf	This document
Block library and example project for TIA Portal V18	109781165_LStream_LIB_v1_6.zip	-

**Note**

The library was additionally tested with a SIMATIC S7-1214C FW 4.4 and a SIMATIC S7-1511 FW 2.8.

## 2 Engineering

### 2.1 Components of the Library

#### Function blocks and functions

Table 2-1: Function blocks and functions of the Library

Name	Version	Description
LStream_JsonDeserializer	V1.6.0	Deserializes a compressed JSON data stream into a sequence of data/objects.
LStream_JsonSerializer	V1.6.0	Serializes a sequence of data/objects into a compressed JSON data stream.
LStream_XmlDeserializer	V1.6.0	Deserializes an XML data stream into a sequence of data/objects.
LStream_XmlSerializer	V1.6.0	Serializes a sequence of data/objects into an XML data stream.
LStream_FindStringIn ByteCharArrayAdv	V1.6.0	Searches an array of bytes for a given string.
LStream_WriteOutString	V1.6.0	Writes a string into the target array of bytes and checks for bound violations

#### PLC Data Types

Table 2-2: PLC library data types

Name	Version	Description
LStream_typeElement	V1.6.0	Data type for storing and reading data/objects. Data is stored as a key-value pair.

## 2.2 Interface description

### 2.2.1 LStream\_JsonDeserializer

#### Description

The block deserializes an array of bytes that corresponds to the compressed JSON data format into a format that can be processed by a SIMATIC S7 Controller.

#### Parameters

Figure 2-1: LStream\_JsonDeserializer

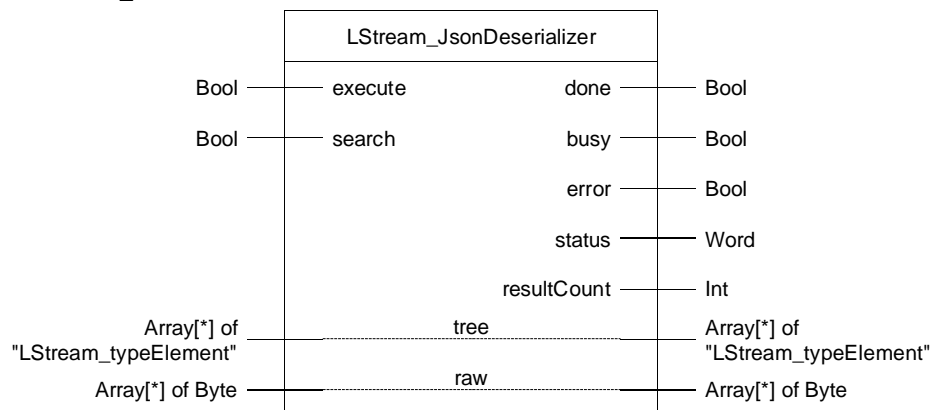


Table 2-3: Parameter "LStream\_JsonDeserializer"

Name	Declaration	Data type	Description
execute	Input	Bool	Rising edge starts deserialization of the JSON data stream.
search	Input	Bool	Only the keys given are deserialized from the JSON data stream.
done	Output	Bool	Job concluded.
busy	Output	Bool	Job is being processed.
error	Output	Bool	An error has occurred in the processing of the FB.
status	Output	Word	Internal status/error code of the FB, see <a href="#">Table 2-4</a> .
resultCount	Output	Int	Number of JSON elements deserialized.
tree	InOut	Array[*] of "LStream_typeElement"	JSON tree containing the deserialized data, see Section <a href="#">2.2.6</a> .
raw	InOut	Array[*] of Byte	Compressed JSON data stream.



## Status and error displays

Table 2-4: Status/error codes

Status	Meaning	Remedy/notes
16#0000	Execution completed without errors.	-
16#7000	No job is currently being processed.	-
16#7001	First call after new job (rising edge at "execute").	-
16#7002	Subsequent call during active processing without further details.	-
16#8201	Error: Array provided is too small.	-
16#8401	Error: No raw data available.	-
16#8600	Error: Error due to an undefined state in the state machine.	-

## Functionality

With a rising edge as the "execute" parameter, the block deserializes the complete array of bytes provided as the "raw" parameter based on the JSON syntax. The result of the deserialization is output as the "tree" parameter.

If the "search" parameter is set, the array of bytes provided as the "raw" parameter is only searched for predefined objects / keys. These keys must be written as the "tree" parameter before the block is executed.

If the FB has completed processing, the output parameter "done" is set and the number of deserialized objects is output as the "resultCount" output.

If an error occurs, the FB aborts processing and sets the output parameter "error" to "TRUE". An error code is output as the output parameter "status".

### Note

Deserialization can take a long time; therefore, the FB works asynchronously (i.e., several call cycles are required for processing). The number of cycles depends on the size of the JSON data stream. With each call, the data stream is processed for the duration defined with the constant MAX\_LOOP\_TIME (3ms by default).

### Note

As soon as the number of deserialized JSON objects exceeds the size of the tree structure — the "tree" parameter — the processing is aborted. All objects that have been deserialized to date are stored in the tree structure.

## 2.2.2 LStream\_JsonSerializer

### Description

The block serializes a given structure into an array of bytes. The array of bytes corresponds to a JSON data stream.

### Parameter

Figure 2-2: LStream\_JsonSerializer

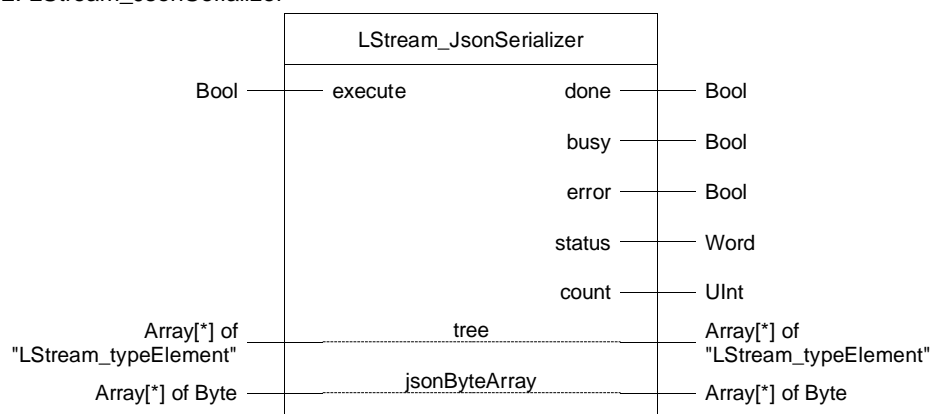


Table 2-5: Parameters of LStream\_JsonSerializer

Name	Declaration	Data type	Description
execute	Input	Bool	Rising edge starts serialization
done	Output	Bool	Serialization completed
busy	Output	Bool	Serialization is being processed
error	Output	Bool	An error has occurred in the processing of the FB
status	Output	Word	Internal status/error code of the FB, see <a href="#">Table 2-6</a> .
count	Output	UInt	Number of byte elements in a Byte Array.
tree	InOut	Array[*] of "LStream_typeElement"	JSON tree containing the serialized data, see Section <a href="#">2.2.6</a> .
jsonByteArray	InOut	Array[*] of Byte	JSON data stream as array of bytes

## Status and error displays

Table 2-6: Status/error codes

Status	Meaning	Remedy/notes
16#0000	Job completed without errors	-
16#7000	No job active	-
16#7001	First call of the command	-
16#7002	Follow-up call of the command (command still running, "busy" = true)	-
16#8201	Error: Array provided is too small.	-
16#8401	Error: Undefined type.	-
16#8402	Error: JSON nesting depth exceeds allowed depth	Constant: Adjust STACK_SIZE
16#8403	Error: JSON depth not specified in tree structure	Specify depth
16#8600	Error: Error due to an undefined state in the state machine.	-
16#8601	Error: Provided tree structure too small	Enlarge tree structure
16#8602	Error: Provided array of bytes too small	Enlarge array
16#8603	Error: Stack too small	See error 16#8402

## Functionality

With a rising edge at the "execute" parameter, the block serializes the array of LStream\_typeElement provided as the "tree" parameter. The JSON data stream is output as a byte array.

The FB terminates serialization as soon as it has reached the end of the array of LStream\_typeElement, or it analyzes an uninitialized element.

When the FB has completed processing, the output parameter "done" is set and the number of bytes written is output as the "count" output.

If an error occurs, the FB aborts processing and sets the output parameter "error" to "TRUE". An error code is output as the output parameter "status".

**Note** Serialization can take a long time; therefore, the FB works asynchronously (i.e., several call cycles are required for processing). The number of cycles depends on the size of the tree structure. With each call, the strings are processed for the duration defined with the constant MAX\_REPEAT\_TIME (3ms by default).

**Note** The hierarchical structure of the JSON must be correctly reflected in the tree structure, starting with the root element with a depth of 1.

**Note** For serialization, the correct type of object must be specified in the tree structure. Permitted types are Element (0), Array (1), String (2), Number (3) and Bool (4); for more information see [Table 2-12](#).

### 2.2.3 LStream\_XmlDeserializer

#### Description

The block deserializes an array of bytes that corresponds to the XML data format into a format that can be processed by a SIMATIC S7 Controller.

#### Parameters

Figure 2-3: LStream\_XmlDeserializer

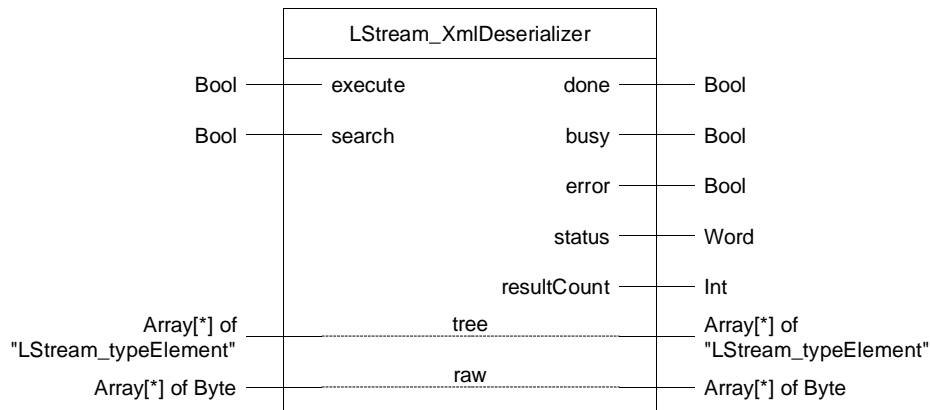


Table 2-7: Parameters of LStream\_XmlDeserializer

Name	Declaration	Data type	Description
execute	Input	Bool	Rising edge starts deserialization of the XML data stream.
search	Input	Bool	Only the keys given are deserialized from the XML data stream.
done	Output	Bool	Job concluded.
busy	Output	Bool	Job is being processed.
error	Output	Bool	An error has occurred in the processing of the FB.
status	Output	Word	Internal status/error code of the FB, see <a href="#">Table 2-8</a> .
resultCount	Output	Int	Number of XML elements deserialized.
tree	InOut	Array[*] of "LStream_typeElement"	XML tree containing the deserialized data, see Section <a href="#">2.2.6</a> .
raw	InOut	Array[*] of Byte	XML data stream.

## Status and error displays

Table 2-8: Status/error codes

Status	Meaning	Remedy/notes
16#0000	Execution completed without errors.	-
16#7000	No job is currently being processed.	-
16#7001	First call after new job (rising edge at "execute").	-
16#7002	Subsequent call during active processing without further details.	-
16#8201	Error: Array provided is too small.	-
16#8401	Error: No raw data available.	-
16#8600	Error: Error due to an undefined state in the state machine.	-

## Functionality

With a rising edge as the "execute" parameter, the block deserializes the complete array of bytes provided as the "raw" parameter based on the XML syntax. The result of the deserialization is output as the "tree" parameter.

If the "search" parameter is set, the array of bytes provided as the "raw" parameter is only searched for predefined objects / keys. These keys must be written as the "tree" parameter before the block is executed.

If the FB has completed processing, the output parameter "done" is set and the number of deserialized objects is output as the "resultCount" output.

If an error occurs, the FB aborts processing and sets the output parameter "error" to "TRUE". An error code is output as the output parameter "status".

### Note

Deserialization can take a long time; therefore, the FB works asynchronously (i.e., several call cycles are required for processing). The number of cycles depends on the size of the XML data stream. With each call, the data stream is processed for the duration defined with the constant MAX\_LOOP\_TIME (3ms by default).

### Note

As soon as the number of deserialized XML objects exceeds the size of the tree structure — the "tree" parameter — the processing is aborted. All objects that have been deserialized to date are stored in the tree structure.

## 2.2.4 LStream\_XmlSerializer

### Description

The block serializes a given structure into an array of bytes. The array of bytes corresponds to an XML data stream.

### Parameters

Figure 2-4: LStream\_XmlSerializer

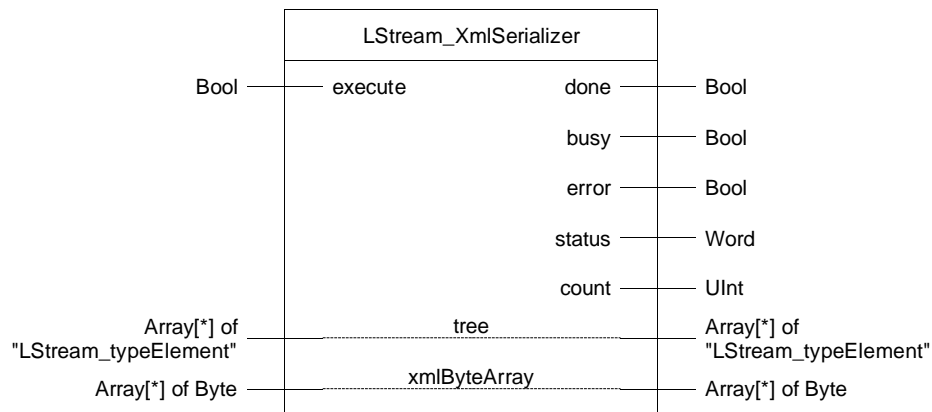


Table 2-9: Parameters of LStream\_XmlSerializer

Name	Declaration	Data type	Description
execute	Input	Bool	Rising edge starts serialization
done	Output	Bool	Serialization completed
busy	Output	Bool	Serialization is being processed
error	Output	Bool	An error has occurred in the processing of the FB
status	Output	Word	Internal status/error code of the FB, see <a href="#">Table 2-10</a> .
count	Output	UInt	Number of byte elements in Byte Array.
tree	InOut	Array[*] of "LStream_typeElement"	XML tree containing the serialized data, see Section <a href="#">2.2.6</a> .
xmlByteArray	InOut	Array[*] of Byte	XML data stream as array of bytes



## Status and error displays

Table 2-10: Status/error codes

Status	Meaning	Remedy/notes
16#0000	Job completed without errors	-
16#7000	No job active	-
16#7001	First call of the command	-
16#7002	Follow-up call of the command (command still running, "busy" = true)	-
16#8201	Error: Array provided is too small.	-
16#8401	Error: Undefined type.	-
16#8402	Error: XML nesting depth exceeds allowed depth	Constant: Adjust STACK_SIZE
16#8403	Error: XML depth not specified in tree structure	Specify depth
16#8600	Error: Error due to an undefined state in the state machine.	-
16#8601	Error: Provided tree structure too small	Enlarge tree structure
16#8602	Error: Provided array of bytes too small	Enlarge array
16#8603	Error: Stack too small	See error 16#8402

## Functionality

With a rising edge at the "execute" parameter, the block serializes the array of LStream\_typeElement provided as the "tree" parameter. The XML data stream is output as a byte array.

The FB terminates serialization as soon as it has reached the end of the array of LStream\_typeElement, or it analyzes an uninitialized element.

When the FB has completed processing, the output parameter "done" is set and the number of bytes written is output as the "count" output.

If an error occurs, the FB aborts processing and sets the output parameter "error" to "TRUE". An error code is output as the output parameter "status".

### Note

Serialization can take a long time; therefore, the FB works asynchronously (i.e., several call cycles are required for processing). The number of cycles depends on the size of the larger tree structure. With each call, the strings are processed for the duration defined with the constant MAX\_REPEAT\_TIME (3ms by default).

### Note

The hierarchical structure of the XML must be correctly reflected in the tree structure, starting with the root element with a depth of 1.

### Note

For serialization, the correct type of object must be specified in the tree structure. Permitted types are element (0), attribute (1); for more information see [Table 2-12](#).

### 2.2.5 LStream\_FindStringInByteArrayAdv

#### Description

The function "LStream\_FindStringInByteArrayAdv" searches an array of bytes for a string and returns the position of the first usage location.

The start index for the search function is passed as the parameter "startPosition".

#### Parameters

Figure 2-5: Function block "LStream\_FindStringInByteArrayAdv"

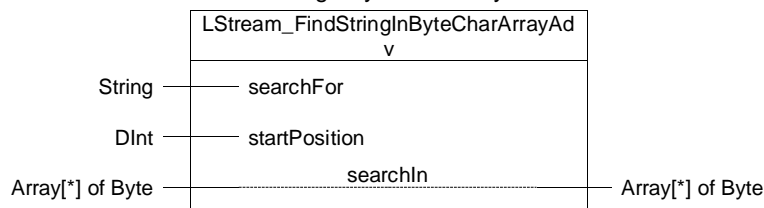


Table 2-11: Parameters of LParse\_FindStringInByteArrayAdv

Name	Declaration	Data type	Description
searchFor	Input	String	Text that is being searched for.
startPosition	Input	DInt	Start position of the search in the array.
searchIn	InOut	Array[*] of Byte	Array of bytes to be searched.
Ret_Val	Return	DInt	Position (index) of the searched string in the array. -1: String was not found.

### 2.2.6 LStream\_typeElement

The PLC data type "LStream\_typeElement" keeps the data/objects of the JSON and XML data streams in a format that can be used by the SIMATIC S7 Controller. The data is held in a key-value pair. In addition, there is the hierarchical depth of the object, as well as its type identifier.

Table 2-12: Parameters of LStream\_typeElement

Name	Data type	Value	Description
type	SInt	-1	Serves as the type identifier of the element. Where -1 is the initial value, which corresponds to an undefined type. When the UDT is used in the context of a JSON structure, 0 represents an object, 1 represents an array, 2 represents a string, 3 represents a number and 4 represents a Bool When the UDT is used in the context of an XML structure, 0 represents an element, 1 represents an attribute
key	String	"	Name of the key
value	String	'NULL'	Name of the key value
depth	SInt	-1	Hierarchical depth of the object
closingElement	Bool	false	The element is last element of an object or array; True=Closing element; Only applicable for JSON functionality

## 2.3 Integration into the user project

### Requirement

The library requires that the data to be deserialized/serialized is available in a data block.

### Integrating the block into the user program

For general information on dealing with libraries in the TIA Portal, see section [3.4](#).

1. Open the "LStream" library in the TIA Portal.
2. Open the folder "Types > LStream" and drag the FB for the desired stream method into the "Program blocks" folder of your PLC. The corresponding functions and PLC data types are automatically copied into your project.
3. Create a DB for controlling and evaluating the FB and open it.
4. At minimum, create the following variables for deserialization:

Figure 2-6: DB "DeserializParam" for controlling and interpreting the FB

DeserializParam			
	Name	Data type	Start value
1	Static		
2	execute	Bool	false
3	search	Bool	false
4	tree	Array[0.."MAX_TREE_SIZE"] of "LStream_typeElement"	

#### Note

Create the variable of the data type "Array[x] of LStream\_typeElement" with sufficient size for your application. Note that the array must start with "0"

5. If you use an FB for serialization, you must, at minimum, create the following variables:

Figure 2-7: DB "SerializParam" for controlling and interpreting the FB

SerializParam			
	Name	Data type	Start value
1	Static		
2	execute	Bool	false
3	xmlByteArray	Array[0.."MAX_BYTE_SIZE"] of Byte	

#### Note

Create the variable of the data type "Array[x]" of bytes with a size sufficient for your application. Note that the array must start with "0"

6. Call the FB at the desired location and create an instance.

7. Connect the parameters of the FB with the variables from the previously created DBs.

Figure 2-8: Calling the FB "LStream\_JsonDeserializer"

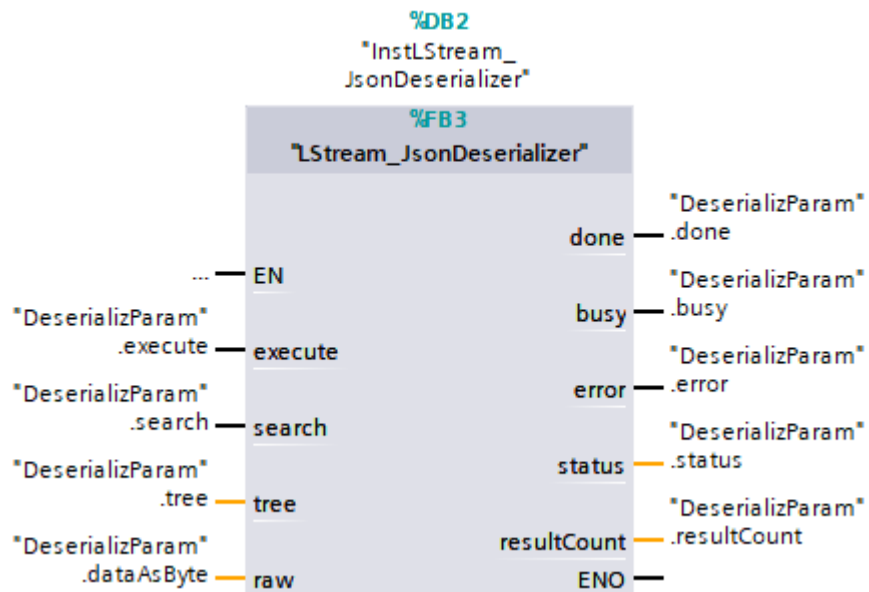


Figure 2-9: Calling the FB "LStream\_JsonSerializer"

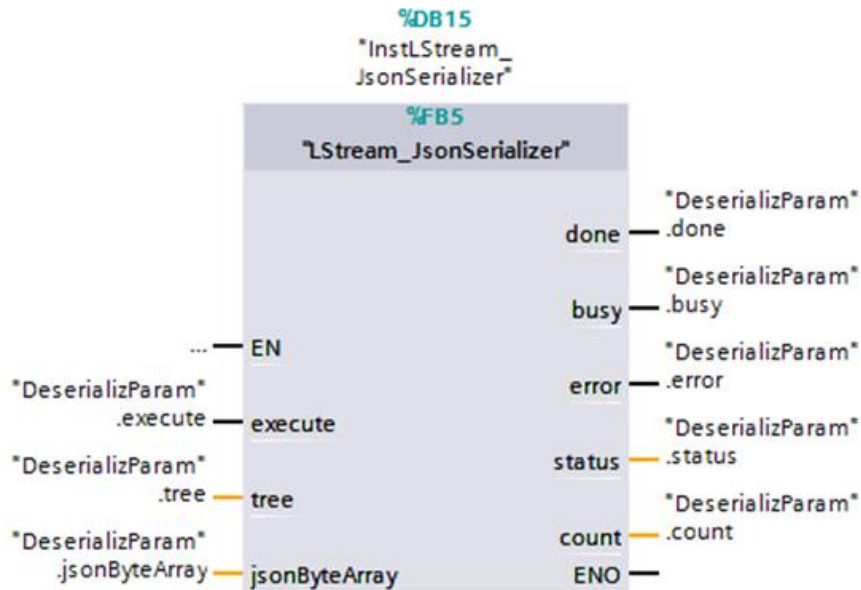


Figure 2-10: Calling the FB "LStream\_XmlDeserializer"

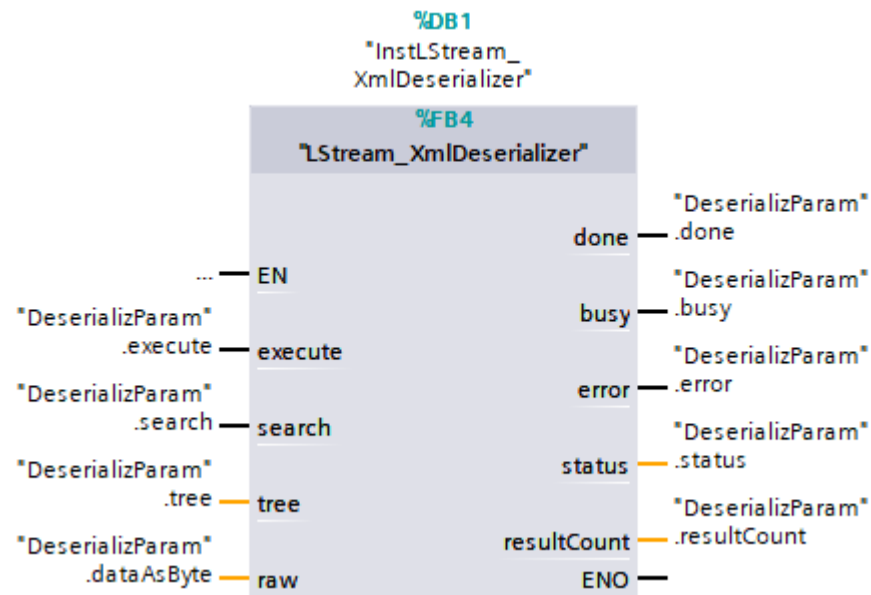
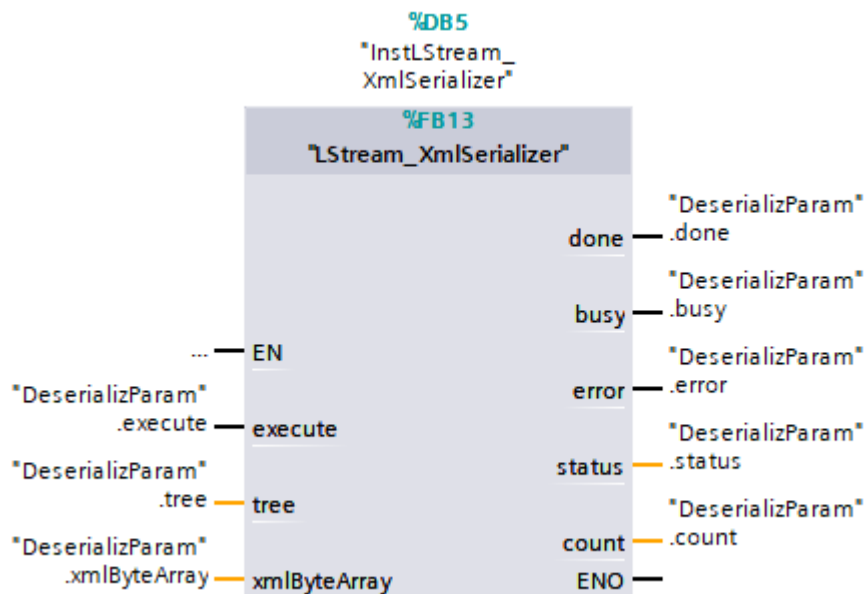


Figure 2-11: Calling the FB "LStream\_XmlSerializer"



8. Load the project into the PLC.

## 2.4 Using & understanding the demo project

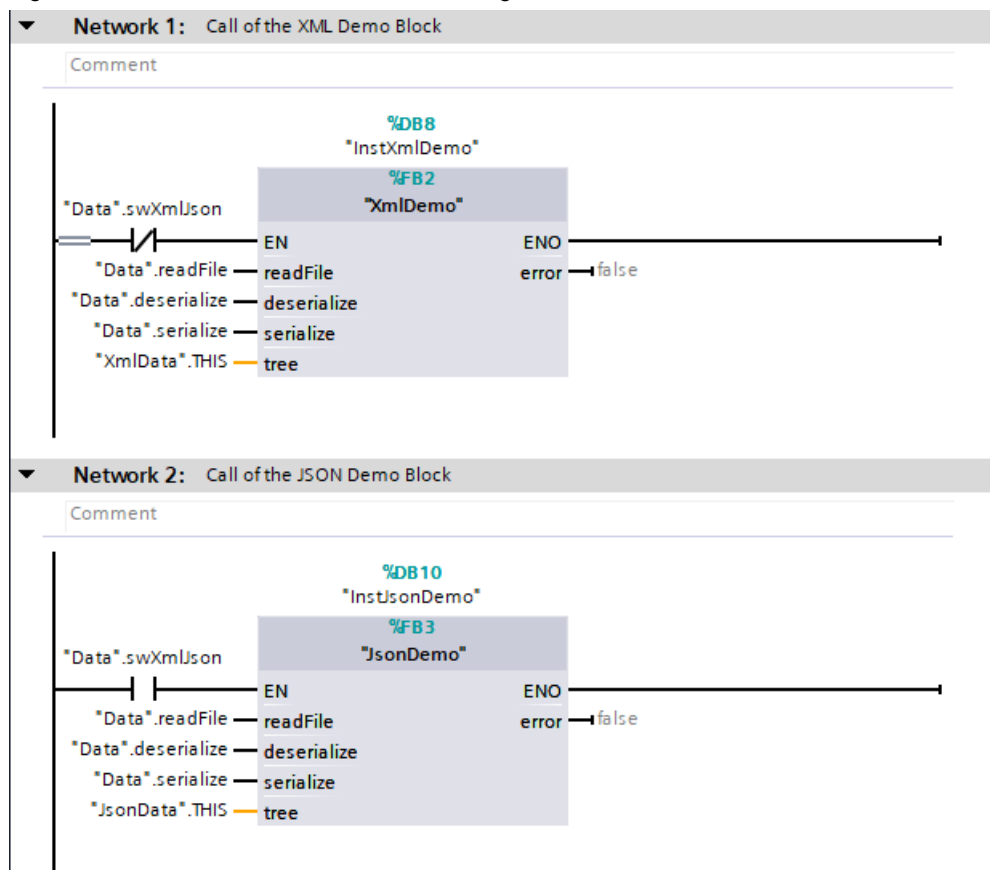
To provide a better understanding of the library and how to create a correct tree for XML or JSON serialization the demo project included in the download shows the call and data structures necessary for a simple structure.

The example files are included in the UserFiles folder of the project and are called demo.json / demo.xml. You must save these files on the SIMATIC Memory Card in the UserFiles folder. There is also a DB for each file format, representing the exact data as the files. These DBs are called JsonData / XmlData.

Usually the workflow would be to read a file (in this case from the webserver), deserialize it into a structure, interpret and use the data in the user program and sometimes sending back a serialized and modified version of the message. In the demo project this is simulated by creating a "round trip" where the result should be the same as the input data.

First the file can be read from the webserver, by setting the readFile in the Data DB to true. Then the byte array can be deserialized into a tree by triggering the deserialize variable. After that the data in the JsonData / XmlData can be used to create a serialized file and write that file to the user files of the Webserver.

Figure 2-12 Screenshot of the Demo User Program





## 3 Additional information

### 3.1 The JavaScript Object Notation (JSON) data exchange format in accordance with RFC 7159

JavaScript Object Notation (JSON) is an easy-to-use, text-based, language-independent data exchange format. It was derived from the ECMAScript Programming Language Standard. JSON defines a small set of formatting rules for portable representation of structured data.

As such, JSON can be used to transfer data in a structured manner between a client and a server application. The format allows for structural evaluation and is readable by both humans and machines.

The JSON format defines 6 structural tokens:

- [ : Left square bracket
- { : Left curly bracket
- ] : Right square bracket
- } : Right curly bracket
- : : Colons
- , : Comma

In addition, insignificant spaces are defined, which are ignored if they are not enclosed in quotation marks ("). These spaces are:

- Space with ASCII code 16#20
- Tab with ASCII code 16#09
- Line feed with ASCII code 16#0A
- Line break (CarriageReturn) with ASCII code 16#0D

There are also 3 literals defined:

- false
- true
- null

### 3.2 JSON Representation in a Tree

Since JSON is a structured format with an easy-to-understand structure, representing it as a list of key-value pairs is the obvious choice. The list is embedded in an array of key-value pair elements. Each of the array elements contains a link to the next list elements. This way, the list does not necessarily have to follow linear array indexing. It allows embedding sub-lists used for the nested JSON substructures such as objects and arrays.

### 3.3 Extensible Markup Language (XML)

Extensible Markup Language (XML) is a simple, very flexible text format derived from SGML (ISO 8879). Originally developed to meet the challenges of large-scale electronic publishing, XML is also playing an increasingly important role in the exchange of a wide variety of data on the web and elsewhere.

For more information about XML, see the following links:

<https://www.w3.org/XML/>

<https://www.w3schools.com/xml>

### 3.4 Library in the TIA Portal

Most of the blocks are stored as types in the library. Thus, the blocks are versioned and can use the following advantages:

- Central update function for library elements
- Versioning of library elements

**Note**

For information on library use in general, see the Guide to Library Use:

<https://support.industry.siemens.com/cs/ww/en/view/109747503>

**Note**

All blocks in the library were created in accordance with the programming style guide:

<https://support.industry.siemens.com/cs/ww/en/view/81318674>

For more information on libraries in the TIA Portal:

- How do you open, edit and upgrade global libraries in the TIA Portal?  
<https://support.industry.siemens.com/cs/ww/en/view/37364723>
- TIA Portal in under 10 minutes: Time Savers – Global libraries  
<https://support.industry.siemens.com/cs/ww/en/view/78529894>
- Which elements from STEP 7 (TIA Portal) and WinCC (TIA Portal) can be stored in a library as a type or as a master copy?  
<https://support.industry.siemens.com/cs/ww/en/view/109476862>
- When starting the TIA Portal V13 and higher, how do you get a global library to open automatically and use it as corporate library, for example?  
<https://support.industry.siemens.com/cs/ww/en/view/100451450>

## 4 Appendix

### 4.1 Service and support

#### Industry Online Support

Do you have any questions or need assistance?

Siemens Industry Online Support offers round the clock access to our entire service and support know-how and portfolio.

The Industry Online Support is the central address for information about our products, solutions and services.

Product information, manuals, downloads, FAQs, application examples and videos – all information is accessible with just a few mouse clicks:

[support.industry.siemens.com](https://support.industry.siemens.com)

#### Technical Support

The Technical Support of Siemens Industry provides you fast and competent support regarding all technical queries with numerous tailor-made offers

– ranging from basic support to individual support contracts. Please send queries to Technical Support via Web form:

[siemens.com/SupportRequest](https://siemens.com/SupportRequest)

#### SITRAIN – Digital Industry Academy

We support you with our globally available training courses for industry with practical experience, innovative learning methods and a concept that's tailored to the customer's specific needs.

For more information on our offered trainings and courses, as well as their locations and dates, refer to our web page:

[siemens.com/sitrain](https://siemens.com/sitrain)

#### Service offer

Our range of services includes the following:

- Plant data services
- Spare parts services
- Repair services
- On-site and maintenance services
- Retrofitting and modernization services
- Service programs and contracts

You can find detailed information on our range of services in the service catalog web page:

[support.industry.siemens.com/cs/sc](https://support.industry.siemens.com/cs/sc)

#### Industry Online Support app

You will receive optimum support wherever you are with the "Siemens Industry Online Support" app. The app is available for iOS and Android:

[support.industry.siemens.com/cs/ww/en/sc/2067](https://support.industry.siemens.com/cs/ww/en/sc/2067)

## 4.2 Industry Mall



The Siemens Industry Mall is the platform on which the entire Siemens Industry product portfolio is accessible. From the selection of products to the order and the delivery tracking, the Industry Mall enables the complete purchasing processing – directly and independently of time and location:

[mall.industry.siemens.com](https://mall.industry.siemens.com)

## 4.3 Links and literature

Table 4-1

Nr.	Thema
\1\	Siemens Industry Online Support <a href="https://support.industry.siemens.com">https://support.industry.siemens.com</a>
\2\	Link to this entry page of this application example <a href="https://support.industry.siemens.com/cs/ww/en/view/109781165">https://support.industry.siemens.com/cs/ww/en/view/109781165</a>
\3\	

## 4.4 Change documentation

Table 4-2

Version	Date	Modifications
V1.0	04/2021	First version
V1.1	11/2021	JSON Deserializer and JSON Serializer added
V1.6	04/2023	Rework and optimizations of JSON & XML functionality