Bibliothek für Daten-Streams (LStream)

TIA Portal, JSON- / XML-Deserializer und JSON- / XML-Serializer

https://support.industry.siemens.com/cs/ww/de/view/109781165

Siemens Industry Online Support

Rechtliche Hinweise

Nutzung der Anwendungsbeispiele

In den Anwendungsbeispielen wird die Lösung von Automatisierungsaufgaben im Zusammenspiel mehrerer Komponenten in Form von Text, Grafiken und/oder Software-Bausteinen beispielhaft dargestellt. Die Anwendungsbeispiele sind ein kostenloser Service der Siemens AG und/oder einer Tochtergesellschaft der Siemens AG („Siemens“). Sie sind unverbindlich und erheben keinen Anspruch auf Vollständigkeit und Funktionsfähigkeit hinsichtlich Konfiguration und Ausstattung. Die Anwendungsbeispiele stellen keine kundenspezifischen Lösungen dar, sondern bieten lediglich Hilfestellung bei typischen Aufgabenstellungen. Sie sind selbst für den sachgemäßen und sicheren Betrieb der Produkte innerhalb der geltenden Vorschriften verantwortlich und müssen dazu die Funktion des jeweiligen Anwendungsbeispiels überprüfen und auf Ihre Anlage individuell anpassen.
Sie erhalten von Siemens das nicht ausschließliche, nicht unterlizenzierbare und nicht übertragbare Recht, die Anwendungsbeispiele durch fachlich geschultes Personal zu nutzen. Jede Änderung an den Anwendungsbeispielen erfolgt auf Ihre Verantwortung. Die Weitergabe an Dritte oder Vervielfältigung der Anwendungsbeispiele oder von Auszügen daraus ist nur in Kombination mit Ihren eigenen Produkten gestattet. Die Anwendungsbeispiele unterliegen nicht zwingend den üblichen Tests und Qualitätsprüfungen eines kostenpflichtigen Produkts, können Funktions- und Leistungsmängel enthalten und mit Fehlern behaftet sein. Sie sind verpflichtet, die Nutzung so zu gestalten, dass eventuelle Fehlfunktionen nicht zu Sachschäden oder der Verletzung von Personen führen.

Haftungsausschluss

Siemens schließt seine Haftung, gleich aus welchem Rechtsgrund, insbesondere für die Verwendbarkeit, Verfügbarkeit, Vollständigkeit und Mangelfreiheit der Anwendungsbeispiele, sowie dazugehöriger Hinweise, Projektierungs- und Leistungsdaten und dadurch verursachte Schäden aus. Dies gilt nicht, soweit Siemens zwingend haftet, z.B. nach dem Produkthaftungsgesetz, in Fällen des Vorsatzes, der groben Fahrlässigkeit, wegen der schuldhaften Verletzung des Lebens, des Körpers oder der Gesundheit, bei Nichteinhaltung einer übernommenen Garantie, wegen des arglistigen Verschweigens eines Mangels oder wegen der schuldhaften Verletzung wesentlicher Vertragspflichten. Der Schadensersatzanspruch für die Verletzung wesentlicher Vertragspflichten ist jedoch auf den vertragstypischen, vorhersehbaren Schaden begrenzt, soweit nicht Vorsatz oder grobe Fahrlässigkeit vorliegen oder wegen der Verletzung des Lebens, des Körpers oder der Gesundheit gehaftet wird. Eine Änderung der Beweislast zu Ihrem Nachteil ist mit den vorstehenden Regelungen nicht verbunden. Von in diesem Zusammenhang bestehenden oder entstehenden Ansprüchen Dritter stellen Sie Siemens frei, soweit Siemens nicht gesetzlich zwingend haftet.
Durch Nutzung der Anwendungsbeispiele erkennen Sie an, dass Siemens über die beschriebene Haftungsregelung hinaus nicht für etwaige Schäden haftbar gemacht werden kann.

Weitere Hinweise

Siemens behält sich das Recht vor, Änderungen an den Anwendungsbeispielen jederzeit ohne Ankündigung durchzuführen. Bei Abweichungen zwischen den Vorschlägen in den Anwendungsbeispielen und anderen Siemens Publikationen, wie z. B. Katalogen, hat der Inhalt der anderen Dokumentation Vorrang.
Ergänzend gelten die Siemens Nutzungsbedingungen (https://support.industry.siemens.com).

Securityhinweise

Siemens bietet Produkte und Lösungen mit Industrial Security-Funktionen an, die den sicheren Betrieb von Anlagen, Systemen, Maschinen und Netzwerken unterstützen.
Um Anlagen, Systeme, Maschinen und Netzwerke gegen Cyber-Bedrohungen zu sichern, ist es erforderlich, ein ganzheitliches Industrial Security-Konzept zu implementieren (und kontinuierlich aufrechtzuerhalten), das dem aktuellen Stand der Technik entspricht. Die Produkte und Lösungen von Siemens formen einen Bestandteil eines solchen Konzepts.
Die Kunden sind dafür verantwortlich, unbefugten Zugriff auf ihre Anlagen, Systeme, Maschinen und Netzwerke zu verhindern. Diese Systeme, Maschinen und Komponenten sollten nur mit dem Unternehmensnetzwerk oder dem Internet verbunden werden, wenn und soweit dies notwendig ist und nur wenn entsprechende Schutzmaßnahmen (z.B. Firewalls und/oder Netzwerksegmentierung) ergriffen wurden.
Weiterführende Informationen zu möglichen Schutzmaßnahmen im Bereich Industrial Security finden Sie unter https://www.siemens.com/industrialsecurity.

Die Produkte und Lösungen von Siemens werden ständig weiterentwickelt, um sie noch sicherer zu machen. Siemens empfiehlt ausdrücklich, Produkt-Updates anzuwenden, sobald sie zur Verfügung stehen und immer nur die aktuellen Produktversionen zu verwenden. Die Verwendung veralteter oder nicht mehr unterstützter Versionen kann das Risiko von Cyber-Bedrohungen erhöhen.
Um stets über Produkt-Updates informiert zu sein, abonnieren Sie den Siemens Industrial Security RSS Feed unter https://www.siemens.com/cert.

Inhaltsverzeichnis

1 Einführung

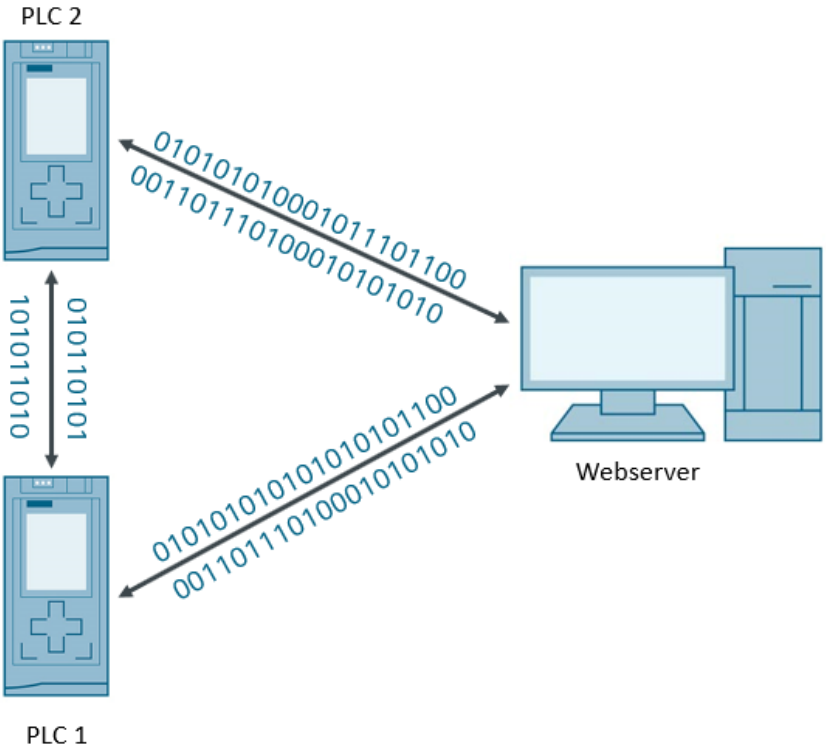
1.1 Überblick

Unter einem Stream versteht man in der Informatik eine Folge von Datenelementen, die im Laufe der Zeit zur Verfügung gestellt werden. Solch eine Abfolge von Datenelementen sind z.B. Datenformate wie JSON oder XML. Diese werden vermehrt für den Austausch von Daten von bzw. zu einem Webserver verwendet.

Durch die zunehmende Vernetzung von Anlagen und das Vorantreiben des Internet der Dinge (IoT), spielt der Austausch von Daten mit dem JSON oder XML-Format in der Automatisierungstechnik eine immer wichtiger werdende Rolle.

Die Bibliothek LStream ermöglicht es die Datenformate JSON und XML in ein für eine SIMATIC S7-Steuerung zur Weiterverarbeitung geeigneteres Format umzuwandeln. Sowie aus abstrakten Daten ein XML-Format zur Weitervermittlung zu erstellen.

Abbildung 1-1: Überblick



1.2 Funktionsweise

Die Bibliothek LStream stellt Funktionsbausteine zur Verfügung, mit denen sich JSON und XML-Datenströme für das Anwenderprogramm deserialisieren und aus dem Anwenderprogramm heraus wieder serialisieren lassen.

1.3 Einschränkungen und Voraussetzungen

Aufgrund von Systemverhalten der SIMATIC PLC und da als Datentypen für Texte und Zeichen aus Speichergründen String/Byte verwendet wurden, ist es nur möglich, ASCII Encoding mit der Bibliothek zu verwenden. Ein anderes Encoding führt zu undefiniertem Verhalten.

Zudem ist die Länge zusammenhängender Texte (z.B. in Values von XML/JSON Elementen) auf 256 Zeichen beschränkt. Diese Einschränkungen lassen sich nicht umgehen; sollte die Funktionalität zwingend benötigt werden, muss die Bibliothek vom Nutzer angepasst werden.

Voraussetzungen JSON

Das JSON muss in einem komprimierten Format vorliegen, das bedeutet:

- Keine Zeilenumbrüche
- Keine Leerzeichen oder Whitespace außerhalb von Werten

Codebeispiel 1-1: Beispiel unkomprimierte JSON-Darstellung

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

Codebeispiel 1-2: Beispiel komprimierte Darstellung

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

Diese Darstellung wurde gewählt, um bei der Deserialisierung und Serialisierung weniger Speicher und Performance zu benötigen.

Die Serialisierung ergibt immer ein komprimiertes Format. Wenn ein unkomprimiertes Format als Eingabedaten für die Deserialisierung gewählt wird ist das Verhalten undefiniert und kann zu fehlerhafter Deserialisierung führen.

1.4 Verwendete Komponenten

Diese Bibliothek wurde mit diesen Hard- und Softwarekomponenten erstellt:

Tabelle 1-3

Komponente	Anzahl	Artikelnummer	Hinweis
SIMATIC CPU 1517TF-3 PN/DP	1	6ES7517-3UP00-0AB0	FW 2.8
SIMATIC STEP 7 Professional V18 Update 1	1	6ES7822-1AA08-0YA5	-

Diese Bibliothek besteht aus den folgenden Komponenten:

Tabelle 1-4

Komponente	Dateiname	Hinweis
Dokumentation	109781165_LStream_DOC_v1_6_de.pdf	Dieses Dokument
Bausteinbibliothek und Beispielprojekt für TIA Portal V18	109781165_LStream_LIB_v1_6.zip	-

Hinweis Die Bibliothek wurde zusätzlich mit einer SIMATIC S7-1214C FW 4.4 und einer SIMATIC S7-1511 FW 2.8 getestet.

2 Engineering

2.1 Bestandteil der Bibliothek

Funktionsbausteine und Funktionen

Tabelle 2-1: Funktionsbausteine und Funktionen der Bibliothek

Name	Version	Beschreibung
LStream_JsonDeserializer	V1.6.0	Deserialisiert einen komprimierten JSON-Datenstrom in eine Sequenz von Daten / Objekten.
LStream_JsonSerializer	V1.6.0	Serialisiert eine Sequenz von Daten / Objekten in einen komprimierten JSON-Datenstrom.
LStream_XmlDeserializer	V1.6.0	Deserialisiert einen XML-Datenstrom in eine Sequenz von Daten / Objekten.
LStream_XmlSerializer	V1.6.0	Serialisiert eine Sequenz von Daten / Objekten in einen XML-Datenstrom.
LStream_FindStringIn ByteCharArrayAdv	V1.6.0	Durchsucht ein Array of Byte nach einer vorgegebenen Zeichenkette.
LStream_WriteOutString	V1.6.0	Schreibt einen String in den Ziel Array of Bytes und prüft dabei auf Überschreitung von Arraygrenzen

PLC-Datentypen

Tabelle 2-2: PLC-Datentypen der Bibliothek

Name	Version	Beschreibung
LStream_typeElement	V1.6.0	Datentyp zum Speichern und auslesen von Daten / Objekten. Daten werden als Key-Value-Pair hinterlegt.

2.2 Schnittstellenbeschreibung

2.2.1 LStream_JsonDeserializer

Beschreibung

Der Baustein deserialisiert ein Array of Byte, das dem komprimierten JSON-Datenformat entspricht, in einer für eine SIMATIC S7-Steuerung verarbeitbares Format.

Parameter

Abbildung 2-1: LStream_JsonDeserializer

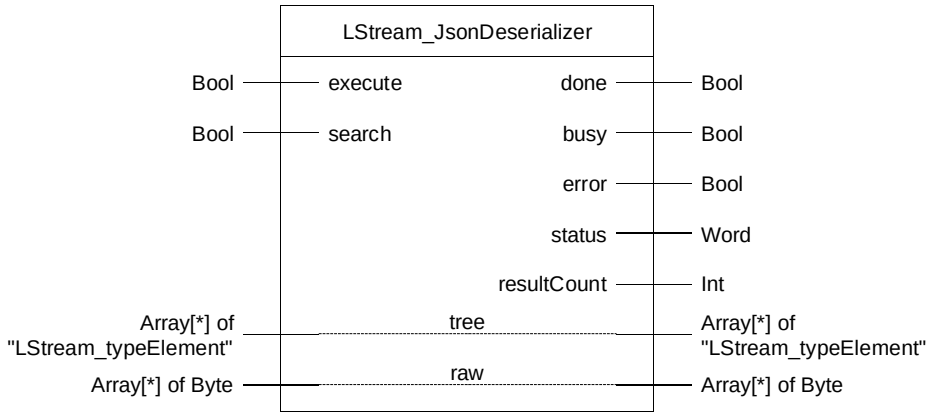


Tabelle 2-3: Parameter "LStream_JsonDeserializer"

Name	Deklaration	Datentyp	Beschreibung
execute	Input	Bool	Steigende Flanke startet das Deserialisieren des bereitgestellten JSON-Datenstroms.
search	Input	Bool	Nur vorgegeben Keys werden aus dem JSON-Datenstrom deserialisiert.
done	Output	Bool	Auftrag abgeschlossen.
busy	Output	Bool	Auftrag wird bearbeitet.
error	Output	Bool	Ein Fehler in der Bearbeitung des FB ist aufgetreten.
status	Output	Word	Interner Status-/Fehlercode des FB, siehe Tabelle 2-4.
resultCount	Output	Int	Anzahl der deserialisierten JSON-Elemente.
tree	InOut	Array[*] of "LStream_typeElement"	JSON-Baum, der die deserialisierten Daten enthält, siehe Kapitel 2.2.6.
raw	InOut	Array[*] of Byte	Komprimierter JSON-Datenstrom.

Status- und Fehleranzeigen

Tabelle 2-4: Status/Fehlercodes

Status	Bedeutung	Abhilfe / Hinweis
16#0000	Ausführung abgeschlossen ohne Fehler.	-
16#7000	Momentan wird kein Auftrag bearbeitet.	-
16#7001	Erster Aufruf nach neuem Auftrag (steigende Flanke an "execute").	-
16#7002	Nachfolgender Aufruf während der aktiven Bearbeitung ohne weitere Details.	-
16#8201	Fehler: bereitgestelltes Array zu klein.	-
16#8401	Fehler: keine Rohdaten vorhanden.	-
16#8600	Fehler: Fehler aufgrund eines undefinierten Zustands in der Zustandsmaschine (state machine).	-

Funktionsweise

Mit einer steigenden Flanke am Parameter "execute" deserialisiert der Baustein das komplette am Parameter "raw" bereitgestellte Array of Byte basierend auf der JSON-Syntax. Das Ergebnis der Deserialisierung wird in dem Parameter "tree" ausgegeben.

Ist der Parameter "search" gesetzt, wird das am Parameter "raw" bereitgestellte Array of Byte nur nach vorgegebenen Objekten / Keys durchsucht. Diese Keys müssen vor der Ausführung des Bausteines in dem Parameter "tree" geschrieben werden.

Hat der FB die Bearbeitung abgeschlossen, wird der Ausgangsparameter "done" gesetzt, und die Anzahl der deserialisierten Objekte wird am Ausgang "resultCount" ausgegeben.

Wenn ein Fehler auftritt, bricht der FB die Bearbeitung ab und setzt den Ausgangsparameter "error" auf "TRUE". Am Ausgangsparameter "status" wird ein Fehlercode ausgegeben.

Hinweis Die Deserialisierung kann viel Zeit in Anspruch nehmen, daher arbeitet der FB asynchron, d. h. für die Bearbeitung werden mehrere Aufrufzyklen benötigt. Die Anzahl der Zyklen hängt von der Größe des JSON-Datenstroms ab. Bei jedem Aufruf wird für die Dauer, definiert mit der Konstanten MAX_LOOP_TIME (Default 3ms), der Datenstrom bearbeitet.

Hinweis Sobald die Anzahl der deserialisierten JSON-Objekte die Größe der Baumstruktur – Parameter "tree" – überschreitet, wird die Bearbeitung abgebrochen. Alle bis dato deserialisierten Objekte sind in der Baumstruktur gespeichert.

2.2.2 LStream_JsonSerializer

Beschreibung

Der Baustein serialisiert eine vorgegebene Struktur in ein Array of Byte. Das Array of Byte entspricht einem JSON-Datenstrom.

Parameter

Abbildung 2-2: LStream_JsonSerializer

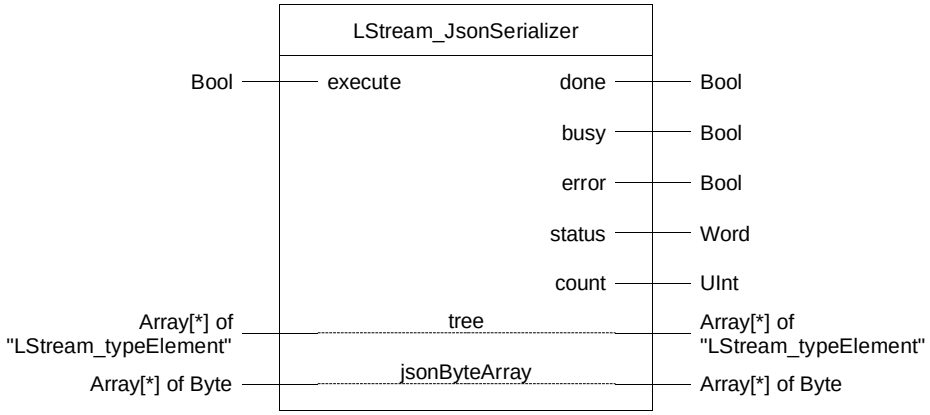


Tabelle 2-5: Parameter of LStream_JsonSerializer

Name	Deklaration	Datentyp	Beschreibung
execute	Input	Bool	Steigende Flanke startet Serialisierung.
done	Output	Bool	Serialisierung abgeschlossen.
busy	Output	Bool	Serialisierung wird bearbeitet.
error	Output	Bool	Ein Fehler in der Bearbeitung des FB ist aufgetreten.
status	Output	Word	Interner Status-/Fehlercode des FB, siehe Tabelle 2-6.
count	Output	UInt	Anzahl der Byte Elemente im Byte Array.
tree	InOut	Array[*] of "LStream_typeElement"	JSON-Baum, der die zu serialisieren Daten enthält, siehe Kapitel 2.2.6.
jsonByteArray	InOut	Array[*] of Byte	JSON-Datenstrom als Array of Byte.

Status- und Fehleranzeigen

Tabelle 2-6: Status/Fehlercodes

Status	Bedeutung	Abhilfe / Hinweis
16#0000	Auftrag ohne Fehler abgeschlossen.	-
16#7000	Kein Auftrag aktiv.	-
16#7001	Erster Aufruf nach neuem Auftrag (steigende Flanke an "execute").	-
16#7002	Folgeaufruf der Anweisung (Anweisung läuft noch, "busy" = TRUE).	-
16#8201	Fehler: bereitgestelltes Array zu klein.	-
16#8401	Fehler: undefinierter Typ.	-
16#8402	Fehler: JSON-Verschachtelungstiefe überschreitet zulässige Tiefe.	Konstante: STACK_SIZE anpassen
16#8403	Fehler: JSON-Tiefe in Baumstruktur nicht angegeben.	Tiefe angeben
16#8600	Fehler: Fehler aufgrund eines undefinierten Zustands in der Zustandsmaschine (state machine).	-
16#8601	Fehler: bereitgestellte Baumstruktur zu klein.	Baumstruktur vergrößern
16#8602	Fehler: bereitgestellte Array of Byte zu klein.	Array vergrößern
16#8603	Fehler: Stack zu klein.	Siehe Fehler 16#8402

Funktionsweise

Mit einer steigenden Flanke am Parameter "execute" serialisiert der Baustein das am Parameter "tree" bereitgestellte Array of LStream_typeElement. Der JSON-Datenstrom wird als Byte Array ausgegeben.

Der FB beendet das Serialisieren, sobald er das Ende des Arrays of LStream_typeElement erreicht hat, oder ein nicht initialisiertes Element analysiert.

Hat der FB die Bearbeitung abgeschlossen, wird der Ausgangsparameter "done" gesetzt und die Anzahl der geschriebenen Bytes wird am Ausgang "count" ausgegeben.

Wenn ein Fehler auftritt, bricht der FB die Bearbeitung ab und setzt den Ausgangsparameter "error" auf "TRUE". Am Ausgangsparameter "status" wird ein Fehlercode ausgegeben.

Hinweis Die Serialisierung kann viel Zeit in Anspruch nehmen, daher arbeitet der FB asynchron, d. h. für die Bearbeitung werden mehrere Aufrufzyklen benötigt. Die Anzahl der Zyklen hängt von der Größe der tree-Struktur ab. Bei jedem Aufruf wird für die Dauer, definiert mit der Konstanten MAX_REPEAT_TIME (Default 3ms), die Zeichenfolgen bearbeitet.

Hinweis Die hierarchische Struktur des JSON muss korrekt in der Baumstruktur wiedergegeben werden. Angefangen beim Root Element mit der Tiefe 1.

Hinweis Für die Serialisierung muss der korrekte Typ des Objektes in der Baumstruktur angegeben werden. Zulässige Typen sind Objekt (0), Array (1), String (2), Nummer (3) und Bool (4). Für mehr Information siehe Tabelle 2-12.

2.2.3 LStream_XmlDeserializer

Beschreibung

Der Baustein deserialisiert ein Array of Byte, das dem XML-Datenformat entspricht, in einer für eine SIMATIC S7-Steuerung verarbeitbares Format.

Parameter

Abbildung 2-3: LStream_XmlDeserializer

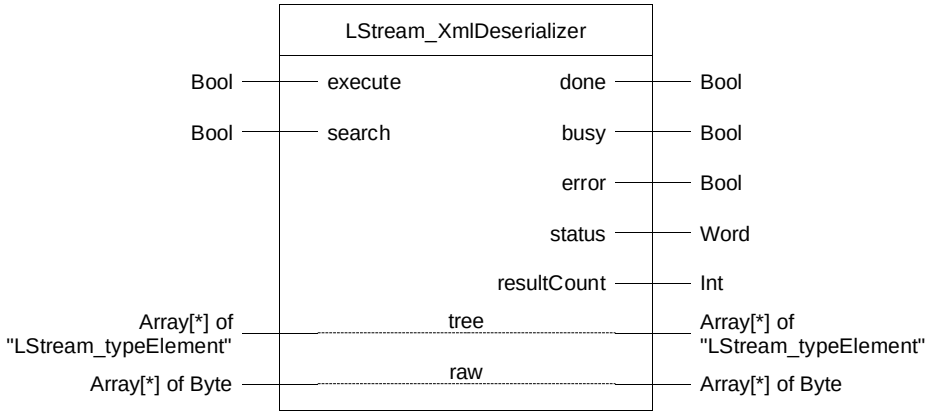


Tabelle 2-7: Parameter of LStream_XmlDeserializer

Name	Deklaration	Datentyp	Beschreibung
execute	Input	Bool	Steigende Flanke startet das Deserialisieren des bereitgestellten XML-Datenstroms.
search	Input	Bool	Nur vorgegeben Keys werden aus dem XML-Datenstrom deserialisiert.
done	Output	Bool	Auftrag abgeschlossen.
busy	Output	Bool	Auftrag wird bearbeitet.
error	Output	Bool	Ein Fehler in der Bearbeitung des FB ist aufgetreten.
status	Output	Word	Interner Status-/Fehlercode des FB, siehe Tabelle 2-8.
resultCount	Output	Int	Anzahl der deserialisierten XML-Elemente.
tree	InOut	Array[*] of "LStream_typeElement"	XML-Baum, der die deserialisierten Daten enthält, siehe Kapitel 2.2.6.
raw	InOut	Array[*] of Byte	XML-Datenstrom.

Status- und Fehleranzeigen

Tabelle 2-8: Status/Fehlercodes

Status	Bedeutung	Abhilfe / Hinweis
16#0000	Ausführung abgeschlossen ohne Fehler.	-
16#7000	Momentan wird kein Auftrag bearbeitet.	-
16#7001	Erster Aufruf nach neuem Auftrag (steigende Flanke an "execute").	-
16#7002	Nachfolgender Aufruf während der aktiven Bearbeitung ohne weitere Details.	-
16#8201	Fehler: bereitgestelltes Array zu klein.	-
16#8401	Fehler: keine Rohdaten vorhanden.	-
16#8600	Fehler: Fehler aufgrund eines undefinierten Zustands in der Zustandsmaschine (state machine).	-

Funktionsweise

Mit einer steigenden Flanke am Parameter "execute" deserialisiert der Baustein das komplette am Parameter "raw" bereitgestellte Array of Byte basierend auf der XML-Syntax. Das Ergebnis der Deserialisierung wird in dem Parameter "tree" ausgegeben.

Ist der Parameter "search" gesetzt, wird das am Parameter "raw" bereitgestellte Array of Byte nur nach vorgegebenen Objekten / Keys durchsucht. Diese Keys müssen vor der Ausführung des Bausteines in dem Parameter "tree" geschrieben werden.

Hat der FB die Bearbeitung abgeschlossen, wird der Ausgangsparameter "done" gesetzt und die Anzahl der deserialisierten Objekte wird am Ausgang "resultCount" ausgegeben.

Wenn ein Fehler auftritt, bricht der FB die Bearbeitung ab und setzt den Ausgangsparameter "error" auf "TRUE". Am Ausgangsparameter "status" wird ein Fehlercode ausgegeben.

Hinweis Die Deserialisierung kann viel Zeit in Anspruch nehmen, daher arbeitet der FB asynchron, d. h. für die Bearbeitung werden mehrere Aufrufzyklen benötigt. Die Anzahl der Zyklen hängt von der Größe des XML-Datenstrom ab. Bei jedem Aufruf wird für die Dauer, definiert mit der Konstanten MAX_LOOP_TIME (Default 3ms), der Datenstrom bearbeitet.

Hinweis Sobald die Anzahl der deserialisierten XML-Objekte die Größe der Baumstruktur – Parameter "tree" – überschreitet, wird die Bearbeitung abgebrochen. Alle bis dato deserialisierten Objekte sind in der Baumstruktur gespeichert.

2.2.4 LStream_XmlSerializer

Beschreibung

Der Baustein serialisiert eine vorgegebene Struktur in ein Array of Byte. Das Array of Byte entspricht einem XML-Datenstrom.

Parameter

Abbildung 2-4: LStream_XmlSerializer

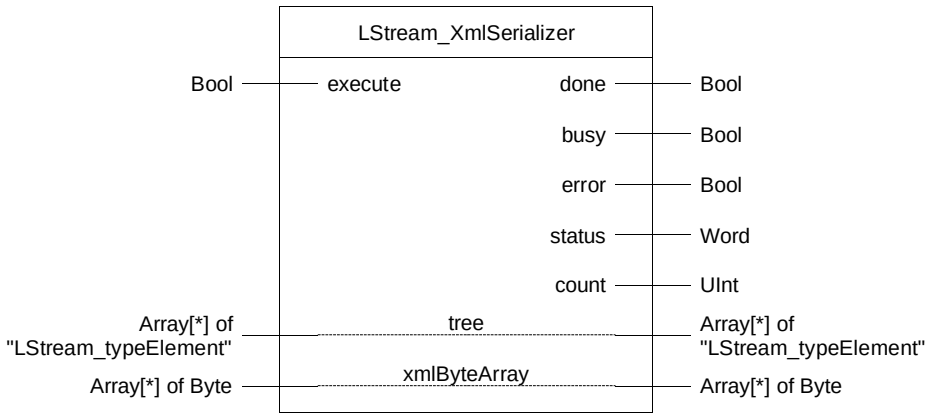


Tabelle 2-9: Parameter of LStream_XmlSerializer

Name	Deklaration	Datentyp	Beschreibung
execute	Input	Bool	Steigende Flanke startet Serialisierung.
done	Output	Bool	Serialisierung abgeschlossen.
busy	Output	Bool	Serialisierung wird bearbeitet.
error	Output	Bool	Ein Fehler in der Bearbeitung des FB ist aufgetreten.
status	Output	Word	Interner Status-/Fehlercode des FB, siehe Tabelle 2-10.
count	Output	UInt	Anzahl der Byte Elemente im Byte Array.
tree	InOut	Array[*] of "LStream_typeElement"	XML-Baum, der die zu serialisieren Daten enthält, siehe Kapitel 2.2.6.
xmlByteArray	InOut	Array[*] of Byte	XML-Datenstrom als Array of Byte.

Status- und Fehleranzeigen

Tabelle 2-10: Status/Fehlercodes

Status	Bedeutung	Abhilfe / Hinweis
16#0000	Auftrag ohne Fehler abgeschlossen.	-
16#7000	Kein Auftrag aktiv.	-
16#7001	Erster Aufruf nach neuem Auftrag (steigende Flanke an "execute").	-
16#7002	Folgeaufruf der Anweisung (Anweisung läuft noch, "busy" = TRUE).	-
16#8201	Fehler: bereitgestelltes Array zu klein.	-
16#8401	Fehler: undefinierter Typ.	-
16#8402	Fehler: XML-Verschachtelungstiefe überschreitet zulässige Tiefe.	Konstante: STACK_SIZE anpassen
16#8403	Fehler: XML-Tiefe in Baumstruktur nicht angegeben.	Tiefe angeben
16#8600	Fehler: Fehler aufgrund eines undefinierten Zustands in der Zustandsmaschine (state machine).	-
16#8601	Fehler: bereitgestellte Baumstruktur zu klein.	Baumstruktur vergrößern
16#8602	Fehler: bereitgestellte Array of Byte zu klein.	Array vergrößern
16#8603	Fehler: Stack zu klein.	Siehe Fehler 16#8402

Funktionsweise

Mit einer steigenden Flanke am Parameter "execute" serialisiert der Baustein das am Parameter "tree" bereitgestellte Array of LStream_typeElement. Der XML-Datenstrom wird als Byte Array ausgegeben.

Der FB beendet das Serialisieren, sobald er das Ende des Arrays of LStream_typeElement erreicht hat, oder ein nicht initialisiertes Element analysiert.

Hat der FB die Bearbeitung abgeschlossen, wird der Ausgangsparameter "done" gesetzt und die Anzahl der geschriebenen Bytes wird am Ausgang "count" ausgegeben.

Wenn ein Fehler auftritt, bricht der FB die Bearbeitung ab und setzt den Ausgangsparameter "error" auf "TRUE". Am Ausgangsparameter "status" wird ein Fehlercode ausgegeben.

Hinweis Die Serialisierung kann viel Zeit in Anspruch nehmen, daher arbeitet der FB asynchron, d. h. für die Bearbeitung werden mehrere Aufrufzyklen benötigt. Die Anzahl der Zyklen hängt von der Größe der tree-Struktur ab. Bei jedem Aufruf wird für die Dauer, definiert mit der Konstanten MAX_REPEAT_TIME (Default 3ms), die Zeichenfolgen bearbeitet.

Hinweis Die hierarchische Struktur des XML muss korrekt in der Baumstruktur wiedergegeben werden. Angefangen beim Root Element mit der Tiefe 1.

Hinweis Für die Serialisierung muss der korrekte Typ des Objektes in der Baumstruktur angegeben werden. Zulässige Typen sind Element (0), Attribut (1).
Für mehr Information siehe Tabelle 2-12.

2.2.5 LStream_FindStringInByteCharArrayAdv

Beschreibung

Die Funktion "LStream_FindStringInByteCharArrayAdv" durchsucht ein Array of Byte nach einem String und gibt die Position der ersten Verwendungsstelle als Rückgabewert aus.

Der Startindex für die Suchfunktion wird mit dem Parameter "startPosition" übergeben.

Parameter

Abbildung 2-5: Funktionsbaustein "LStream_FindStringInByteCharArrayAdv"

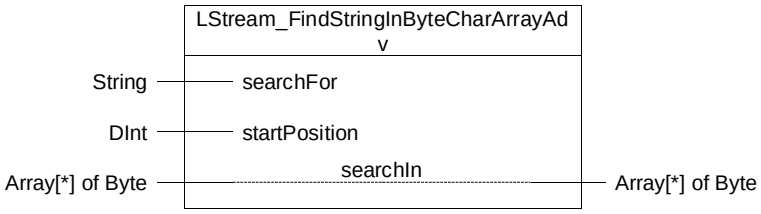


Tabelle 2-11: Parameter of LStream_FindStringInByteCharArrayAdv

Name	Deklaration	Datentyp	Beschreibung
searchFor	Input	String	Text, nach dem gesucht wird.
startPosition	Input	DInt	Startposition der Suche im Array.
searchIn	InOut	Array[*] of Byte	Array of Byte das durchsucht werden soll.
Ret_Val	Return	DInt	Position (Index) des gesuchten Strings im Array. -1: String wurde nicht gefunden.

2.2.6 LStream_typeElement

Der PLC-Datentyp "LStream_typeElement" hält die Daten / Objekte der JSON- und XML-Datenströme in einem für die SIMATIC S7-Steuerung verwertbaren Format. Die Daten werden in einem Key-Value-Pair gehalten. Hinzu kommt die hierarchische Tiefe des Objektes, sowie seine Typkennung.

Tabelle 2-12: Parameter of LStream_typeElement

Name	Datentyp	Wert	Beschreibung
type	SInt	-1	Dient der Typkennung des Elementes. Wobei -1 der Startwert ist, der einen undefinierten Typ darstellt. Wird der UDT im Zusammenhang mit einer JSON-Struktur verwendet, repräsentiert 0 ein Objekt, 1 repräsentiert ein Array, 2 repräsentiert ein String, 3 repräsentiert eine Nummer und 4 repräsentiert ein Bool. Wird der UDT im Zusammenhang mit einer XML-Struktur verwendet, repräsentiert 0 ein Element, 1 repräsentiert ein Attribut.
key	String	''	Name des Schlüssels
value	String	'NULL'	Name des Schlüsselwertes
depth	SInt	-1	Hierarchische Tiefe des Objektes
closingElement	Bool	FALSE	Information für den Serializer, dass das aktuelle Element der letzte Eintrag eines Arrays ist und dieses geschlossen werden kann.

2.3 Integration ins Anwenderprojekt

Voraussetzung

Die Bibliothek setzt voraus, dass die zu deserialisierenden/serialisierenden Daten in einem Datenbaustein bereitstehen.

Baustein ins Anwenderprogramm integrieren

Allgemeine Informationen zum Umgang mit Bibliotheken im TIA Portal finden Sie in Kapitel 3.4.

1. Öffnen Sie die Bibliothek "LStream" im TIA Portal.
2. Öffnen Sie den Ordner "Typen > LStream" und ziehen Sie den FB für die gewünschte Stream-Methode in den Ordner "Programmbausteine" ("Program blocks") Ihrer PLC. Die zugehörigen Funktionen und PLC-Datentypen werden automatisch in Ihr Projekt kopiert.
3. Legen Sie einen DB zum Steuern und Auswerten des FB an und öffnen Sie diesen.
4. Legen Sie zum Deserialisieren mindestens die folgenden Variablen an:

Abbildung 2-6: DB "DeserializParam" zum Steuern und Auswerten des 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"	

Hinweis Legen Sie die Variable vom Datentyp "Array[0..x] of LStream_typeElement" mit für Ihre Anwendung ausreichender Größe an. Beachten Sie, dass das Array mit "0" beginnen muss.

5. Wenn sie einen FB zum Serialisieren verwenden, legen Sie mindestens die folgenden Variablen an:

Abbildung 2-7: DB "SerializParam" zum Steuern und Auswerten des FB

SerializParam

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

Hinweis Legen Sie die Variable vom Datentyp "Array[0..x] of Byte mit für Ihre Anwendung ausreichender Größe an. Beachten Sie, dass das Array mit "0" beginnen muss.

6. Rufen Sie den FB an der gewünschten Stelle auf und erstellen Sie eine Instanz.

7. Verschalten Sie die Parameter des FB mit den Variablen aus den zuvor angelegten DBs.

Abbildung 2-8: Aufruf des FB "LStream_JsonDeserializer"

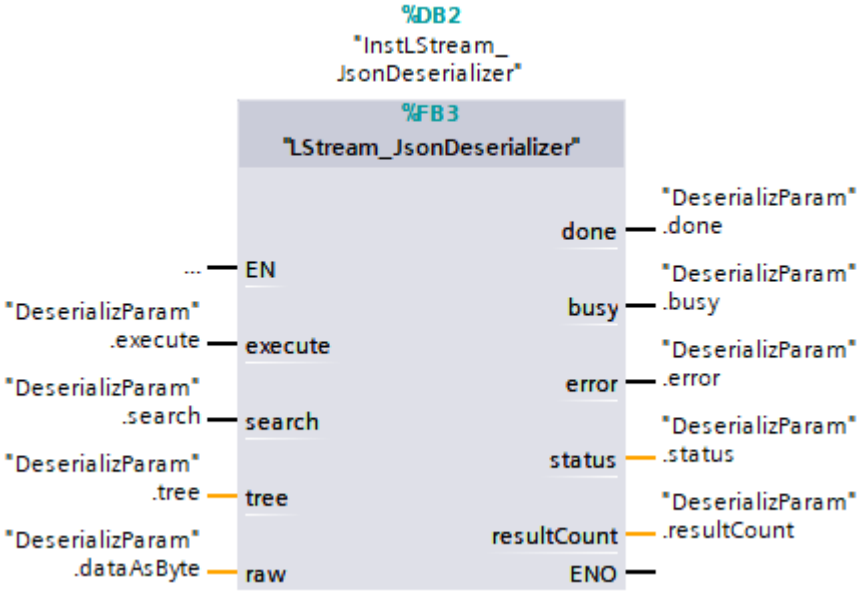


Abbildung 2-9: Aufruf des FB "LStream_JsonSerializer"

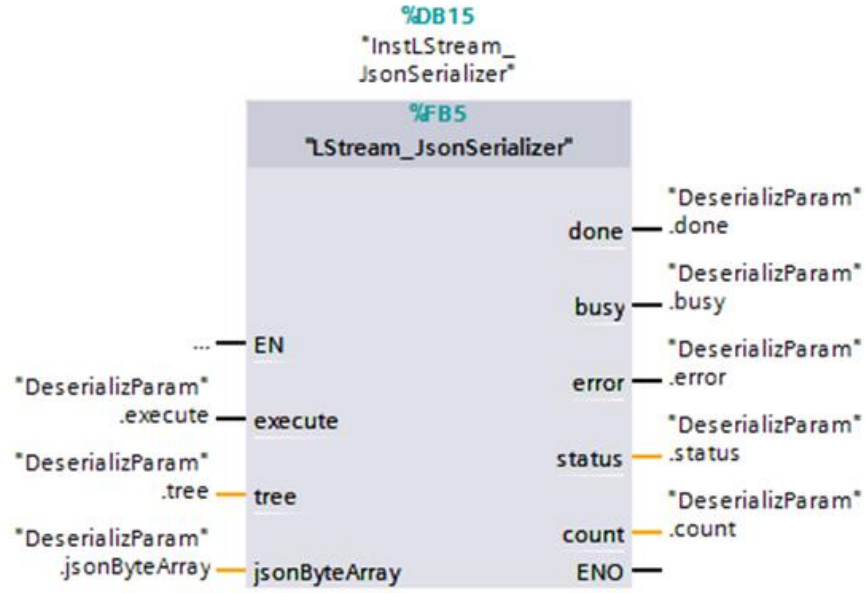


Abbildung 2-10: Aufruf des FB "LStream_XmlDeserializer"

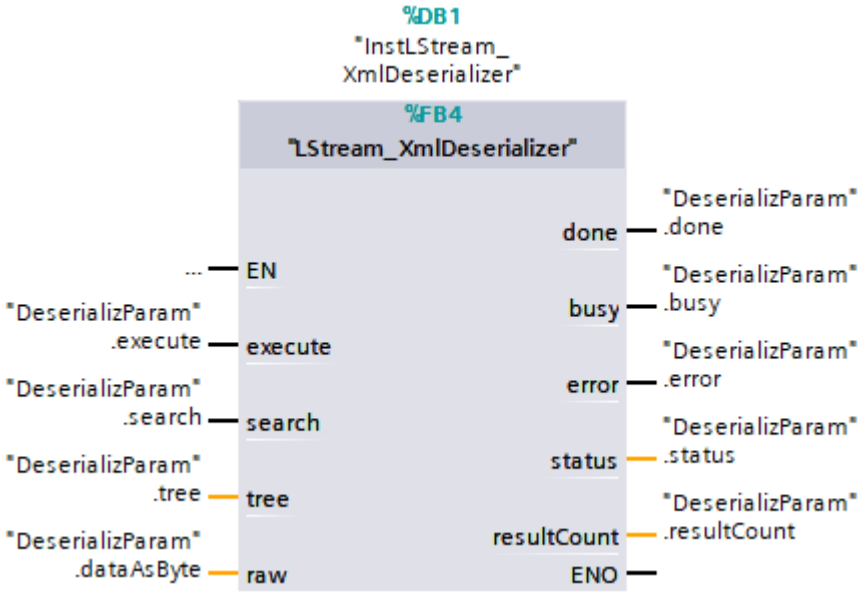
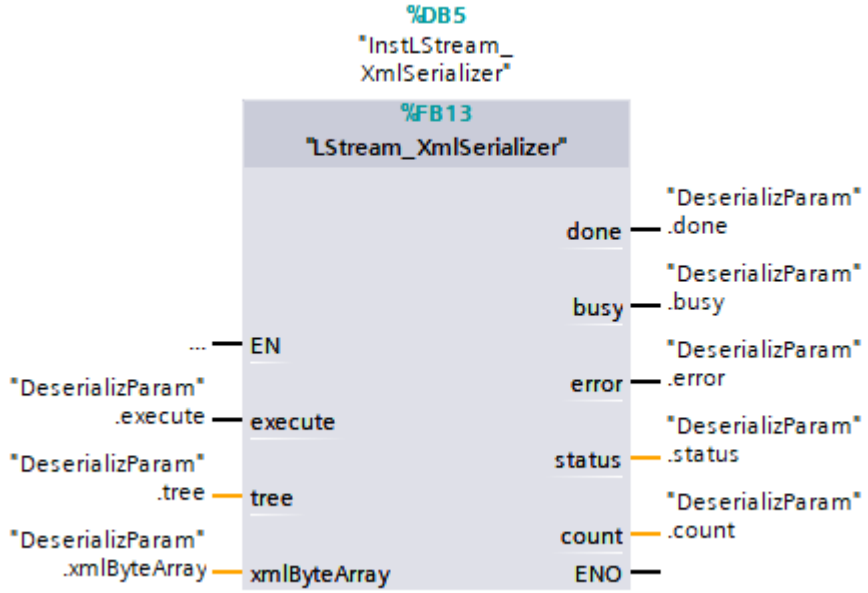


Abbildung 2-11: Aufruf des FB "LStream_XmlSerializer"



8. Laden Sie das Projekt in die PLC.

2.4 Verwendung des Demo Projektes

Um ein besseres Verständnis für die Funktion und Verwendung der Bibliothek zu bekommen ist in dem Download der Library auch ein Demoprojekt enthalten. Dieses zeigt die Verwendung der Deserialisierung und Serialisierung anhand einer kleinen, aber komplexen Datenstruktur.

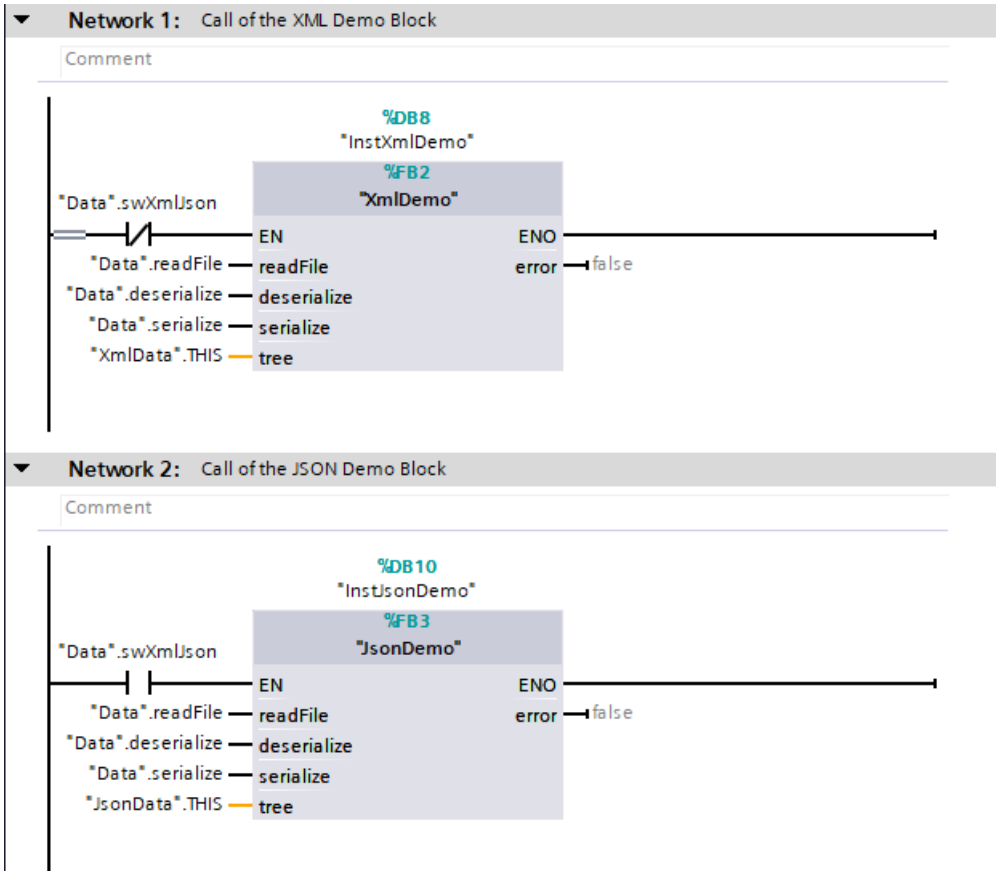
Die Beispieldateien sind im Projektverzeichnis im Ordner UserFiles enthalten. Diese Dateien müssen Sie auf der SIMATIC Memory Card im Ordner UserFiles speichern. Im Projekt finden Sie in den DBs JsonData bzw. XmlData die Abbildung der Information als Array of LStream_typeElement.

Normalerweise ist die Anforderung bzw. Verwendung von De-/Serialisierung eine bestimmte Datei/ einen Datenstream einzulesen, diesen zu deserialisieren, um die Daten im Nutzer-programm zu interpretieren und Werte auszulesen bzw. zu verändern. Ist dies geschehen, sollten die Daten wieder serialisiert werden, um das resultierende Format als Datei / Datenstream weiterzugeben.

Dieser Workflow ist im Demo Projekt anhand eines "Round-Trips", bei dem eine Datei (in diesem Fall vom Webserver) eingelesen wird und nach einer Deserialisierung und erneuten Serialisierung identisch mit den ursprünglichen Informationen sein sollte.

Der Demo-Workflow kann gestartet werden, indem die readFile Variable im Data DB auf TRUE gesetzt wird. Der resultierende Byte Array kann mit der Variable "deserialize" in die Array of LStream_typeElement deserialisiert werden. Dies modifiziert die JsonData/XmlData DBs, die jetzt mit der Variable "serialize" wieder als Datei für den Webserver serialisiert werden kann.

Abbildung 2-12 Screenshot aus dem Demo User Programm



3 Wissenswertes

3.1 Das JavaScript Object Notation (JSON) Datenaustauschformat gemäß RFC 7159

JavaScript Object Notation (JSON) ist ein leichtes, textbasiertes, sprachunabhängiges Datenaustauschformat. Es wurde vom ECMAScript Programming Language Standard abgeleitet. JSON definiert einen kleinen Satz von Formatierungsregeln für die tragbare Darstellung strukturierter Daten.

Als solches kann JSON verwendet werden, um Daten strukturiert zwischen einem Client und einer Serveranwendung zu übertragen. Das Format ermöglicht eine Strukturbewertung und ist sowohl für Menschen als auch für Maschinen lesbar.

Das JSON-Format definiert 6 strukturelle Token:

- [: Linke eckige Klammer
- { : Linke geschweifte Klammer
-] : Rechte eckige Klammer
- } : Rechte geschweifte Klammer
- : : Doppelpunkt
- , : Komma

Außerdem sind unbedeutende Leerzeichen definiert, die ignoriert werden, wenn sie nicht in Anführungszeichen (") gesetzt sind. Diese Leerzeichen sind:

- Leerzeichen (space) mit ASCII code 16#20
- Tab mit ASCII code 16#09
- Zeilenvorschub (LineFeed) mit ASCII code 16#0A
- Zeilenumbruch (CarriageReturn) mit ASCII code 16#0D

Es sind auch 3 Literale definiert:

- FALSE
- TRUE
- NULL

3.2 JSON-Darstellung in einem Baum

Da es sich bei JSON um ein strukturiertes Format mit einer einfach zu verstehenden Struktur handelt, ist die Darstellung als Liste von Schlüssel-Wert-Paaren (Key-Value-Pairs) die naheliegende Wahl. Die Liste ist in ein Array von Key-Value-Pair-Elementen eingebettet. Jedes der Array-Elemente enthält einen Link zu den nächsten Listenelementen. Auf diese Weise muss die Liste nicht unbedingt der linearen Array-Indizierung folgen. Sie ermöglicht die Einbettung von Teillisten, die für die verschachtelten JSON-Unterstrukturen wie Objekte und Arrays verwendet werden.

3.3 Extensible Markup Language (XML)

Extensible Markup Language (XML) ist ein einfaches, sehr flexibles Textformat, das von SGML (ISO 8879) abgeleitet ist. Ursprünglich für die Herausforderungen des elektronischen Publizierens in großem Maßstab entwickelt, spielt XML auch beim Austausch verschiedenster Daten im Web und anderswo eine immer wichtigere Rolle.

Weiter Informationen zu XML finden Sie unter folgenden Links:

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

https://www.w3schools.com/xml

3.4 Bibliotheken im TIA Portal

Der Großteil der Bausteine ist als Typ in der Bibliothek abgelegt. Somit sind die Bausteine versioniert und können folgende Vorteile nutzen:

- Zentrale Updatefunktion von Bibliothekselementen
- Versionierung von Bibliothekselementen

Hinweis Informationen zum generellen Umgang mit Bibliotheken finden Sie im Leitfaden zur Bibliothekshandhabung:

https://support.industry.siemens.com/cs/ww/de/view/109747503

Hinweis Alle Bausteine in der Bibliothek wurden nach dem Programmierstyleguide erstellt:

https://support.industry.siemens.com/cs/ww/de/view/81318674

Weitere Informationen zu Bibliotheken im TIA Portal:

- Wie können Sie globale Bibliotheken im TIA Portal öffnen, bearbeiten und hochrüsten?
  https://support.industry.siemens.com/cs/ww/de/view/37364723
- In weniger als 10 Minuten TIA Portal: Time Savers – Globale Bibliotheken
  https://support.industry.siemens.com/cs/ww/de/view/78529894
- Welche Elemente aus STEP 7 (TIA Portal) und WinCC (TIA Portal) können in einer Bibliothek als Typ oder als Kopiervorlage abgelegt werden?
  https://support.industry.siemens.com/cs/ww/de/view/109476862
- Wie können Sie beim Starten von TIA Portal ab V13 eine globale Bibliothek automatisch öffnen und z. B. als Unternehmensbibliothek verwenden?
  https://support.industry.siemens.com/cs/ww/de/view/100451450

4 Anhang

4.1 Service und Support

Industry Online Support

Sie haben Fragen oder brauchen Unterstützung?

Über den Industry Online Support greifen Sie rund um die Uhr auf das gesamte Service und Support Know-how sowie auf unsere Dienstleistungen zu.

Der Industry Online Support ist die zentrale Adresse für Informationen zu unseren Produkten, Lösungen und Services.

Produktinformationen, Handbücher, Downloads, FAQs und Anwendungsbeispiele – alle Informationen sind mit wenigen Mausklicks erreichbar:

support.industry.siemens.com

Technical Support

Der Technical Support von Siemens Industry unterstützt Sie schnell und kompetent bei allen technischen Anfragen mit einer Vielzahl maßgeschneiderter Angebote
– von der Basisunterstützung bis hin zu individuellen Supportverträgen.

Anfragen an den Technical Support stellen Sie per Web-Formular:

siemens.com/SupportRequest

SITRAIN – Digital Industry Academy

Mit unseren weltweit verfügbaren Trainings für unsere Produkte und Lösungen unterstützen wir Sie praxisnah, mit innovativen Lernmethoden und mit einem kundenspezifisch abgestimmten Konzept.

Mehr zu den angebotenen Trainings und Kursen sowie deren Standorte und Termine erfahren Sie unter:

siemens.de/sitrain

Serviceangebot

Unser Serviceangebot umfasst folgendes:

- Plant Data Services
- Ersatzteilservices
- Reparaturservices
- Vor-Ort und Instandhaltungsservices
- Retrofit- und Modernisierungsservices
- Serviceprogramme und Verträge

Ausführliche Informationen zu unserem Serviceangebot finden Sie im Servicekatalog:

support.industry.siemens.com/cs/sc

Industry Online Support App

Mit der App "Siemens Industry Online Support" erhalten Sie auch unterwegs die optimale Unterstützung. Die App ist für iOS und Android verfügbar:

support.industry.siemens.com/cs/ww/de/sc/2067

4.2 Industry Mall

Die Siemens Industry Mall ist die Plattform, auf der das gesamte Produktportfolio von Siemens Industry zugänglich ist. Von der Auswahl der Produkte über die Bestellung und die Lieferverfolgung ermöglicht die Industry Mall die komplette Einkaufsabwicklung – direkt und unabhängig von Zeit und Ort:

mall.industry.siemens.com

4.3 Links und Literatur

Tabelle 4-1

Nr.	Thema
\1\	Siemens Industry Online Support https://support.industry.siemens.com
\2\	Link auf die Beitragsseite des Anwendungsbeispiels https://support.industry.siemens.com/cs/ww/de/view/109781165
\3\	

4.4 Änderungsdokumentation

Tabelle 4-2

Version	Datum	Änderung
V1.0	04/2021	Erste Ausgabe
V1.1	11/2021	JSON Deserializer und JSON Serializer ergänzt
V1.6	04/2023	Überarbeitung und Optimierungen der JSON/XML Funktionalität