Inhaltsverzeichnis

Konfiguration 1
Channel 1
Settings 1
Hinzufügen eines Channels 2
Variablen 3
Path 4
Beispiele 4
Fehlerdiagnose 5
Kanal 6
Variablen 6
Logdatei 7
Entities 7
Device 7
Channel 8
Ordner & Dateien 8
Ordner 8
Dateien 8
Versionsinformation 8
Dieses Dokument 8
Plugin 9
Assembly 9

Allen-Bradley Device Plugin

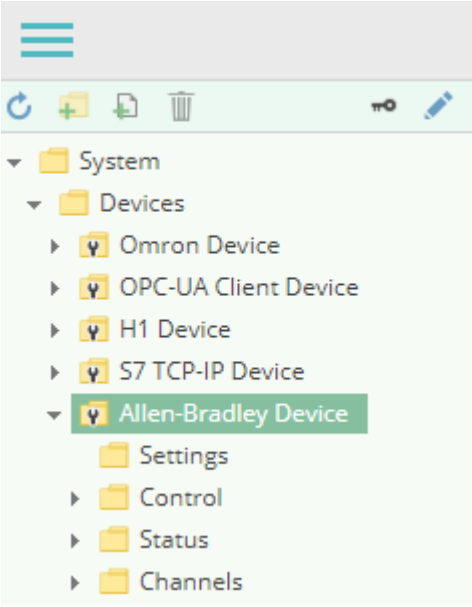
Das Allen-Bradley Device Plugin ermöglicht das Lesen und Schreiben von Daten von Allen-Bradley-SPS-Geräten über EtherNet/IP.

Folgende Gerätetypen werden unterstützt:

- ControlLogix/CompactLogix
- Micro800
- MicroLogix
- PLC-5
- SLC 500

Konfiguration

Die gesamte Allen-Bradley Device Plugin-Konfiguration befindet sich unter dem Nodepfad `/System/Devices/Allen-Bradley Device`.



Channel

Ein Allen-Bradley Device Channel repräsentiert die Verbindung zu einer Allen-Bradley SPS.

Settings

Address

IP-Adresse der Allen-Bradley-SPS.

Device Type

Der Gerätetyp der Allen-Bradley-SPS. Es werden folgende Gerätetypen unterstützt:

- ControlLogix/CompactLogix
- Micro800
- MicroLogix
- PLC-5
- SLC 500

CIP Path

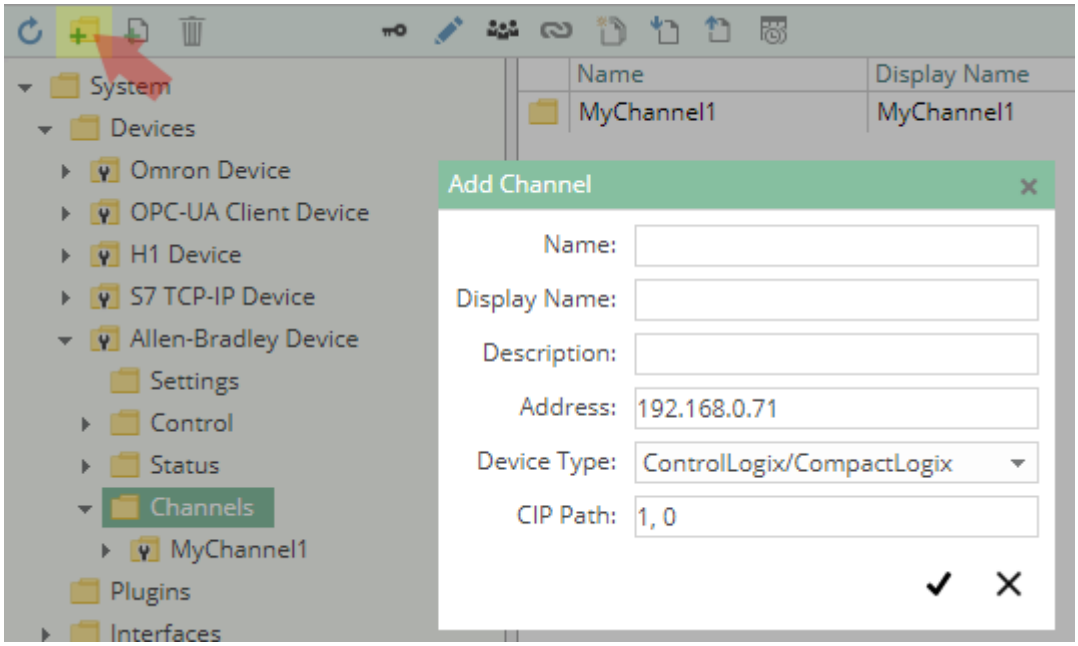
Der CIP-Pfad zur SPS. Dieses Feld muss angegeben werden bei ControlLogix/CompactLogix sowie bei Verwendung einer DH+-Protokollbrücke (d.h. ein DHRIO-Modul).
Dieses Feld kann das Format `A,B` haben, wobei `A` den Porttyp angibt (`1`=Backplane), und `B` den Slot angibt, in dem die CPU gesteckt ist.

Beispiel: `1,0`

Operation Timeout

Das Timeout, das bei Lese- und Schreiboperationen angewendet werden soll.

Hinzufügen eines Channels



Um einen neuen Allen-Bradley-Channel zu erstellen, gehen Sie wie folgt vor:

1. Fügen Sie einen Folder Node unter dem Node `Allen-Bradley Device/Channels` hinzu, oder machen Sie einen Rechtsklick auf den `Allen-Bradley Device/Channels`-Node und wählen Sie `Add Channel` aus.
2. Tragen Sie im `Add Channel`-Dialog die Settings für die Allen-Bradley-Verbindung ein.
3. Nachdem Sie „Save“ geklickt haben, wird die Channel-Node erstellt.
4. Sie können den Kanal starten, indem Sie die Channel-Node auswählen und den Startbutton klicken.

Variablen

Unter dem `Variables`-Node können Sie Datenpunktnodes erstellen, die aus der Allen-Bradley-SPS als **Tags** gelesen und in diese geschrieben werden.
Die `Value Type`-Eigenschaft muss dabei auf den entsprechenden Typen des Tags festgelegt werden. Aktuell werden folgenden Tag-Typen unterstützt:

Tag-Typ	**CoDaBix-Typ**
`SINT`	`SByte` bzw. `SByte-Array`
`USINT`	`Byte` bzw. `Byte-Array`
`INT`	`Int16` bzw. `Int16-Array`
`UINT`	`UInt16` bzw. `UInt16-Array`
`DINT`	`Int32` bzw. `Int32-Array`
`UDINT`	`UInt32` bzw. `UInt32-Array`
`LINT`	`Int64` bzw. `Int64-Array`
`ULINT`	`UInt64` bzw. `UInt64-Array`
`REAL`	`Single` bzw. `Single-Array`
`STRING` (oder benutzerdefinierter String-Datentyp)	`String` bzw. `String-Array`

Bei **String**-Datentypen wird ein ASCII/ISO-8859-1-ähnliches Encoding angenommen (das High Byte des 16-Bit-Chars wird abgeschnitten). Dabei werden zwar Surrogate-Pairs (für Characters außerhalb

der Unicode-BMP) nicht korrekt behandelt, jedoch ist damit die Anzahl der Bytes gleich der Länge des Strings.

Beachten Sie: Das jeweilige signierte/unsignierte Pendant eines Datentyps kann auch anstelle dessen verwendet werden. Z.B. kann ein `SINT` auch als `Byte` statt als `SByte` verwendet werden; jedoch wird das MSB (most significant bit) dann anders interpretiert, da dieses bei einem signierten Datentypen das Vorzeichenbit ist.

Path

Über die `Path`-Eigenschaft des Nodes wird der Name des Tags, optional eine Typangabe, sowie (bei Arrays) optional ein Offset und die Länge der Daten definiert:

```
<TagName>
<TagName>, <Length>
<TagName>, <Type>
<TagName>, <Type>[<Length>]
```

Platzhalter	Beschreibung
`<TagName>`	Der vollständige Name des Tags. Falls das Tag ein Programm-Tag ist, muss als Präfix `Program:<Programmname>.` mit dem entsprechenden Programmnamen verwendet werden. Falls das Tag eine Struktur ist, kann nach einem Punkt (`.`) der Name des zu verwendenden Feldes angegeben werden. Falls das Tag (oder ein Struktur-Feld) ein Array ist, kann mit der Syntax `[Offset]` das Array-Offset angegeben werden.
`<Length>`	Falls das Tag ein Array ist, kann hier die Länge des Arrays angegeben werden. Diese wird nur beim Lesen verwendet; beim Schreiben wird die Länge aus dem zu schreibenden Array ermittelt.
`<Type>`	Falls vorhanden, gibt eine Typspezifikation an, wenn diese nicht aus dem CoDaBix-Typen abgeleitet werden kann. Aktuell können folgende Typen angegeben werden: • `STRING:<MaxLen>`: Gibt die maximale Anzahl an String-Characters an, wenn die Variable vom Typ `String` bzw. `String-Array` ist. Dies kann bei benutzerdefinierten String-Datentypen verwendet werden; ansonsten wird der vordefinierte `STRING`-Datentyp mit max. 82 Characters verwendet.

Beispiele

Value Type	Path	Erklärung
`Int16`	`Local:1:O.Data`	`Data`-Feld (`INT`) des digitalen Ausgangs (`Int16` enthält Bitmaske der 16 digitalen Ausgänge (Digital Output Points))
`Int16`	`Local:1:I.Data`	`Data`-Feld (`INT`) des digitalen Eingangs (`Int16` enthält Bitmaske der 16 digitalen Eingänge (Digital Input Points))

Value Type	Path	Erklärung
Int32	MyControllerTag	Controller-Tag MyControllerTag (DINT)
Int32	Program:MainProgram.MyTag1.MyField1	Vom Programm MainProgram das Tag MyTag1 (Struktur) mit dem Feld MyField1 (DINT)
Byte-Array	Program:MainProgram.MyTag2, 20	Vom Programm MainProgram das Tag MyTag2 (SINT[20]) mit Länge 20 (Bereich 0..20)
Byte-Array	Program:MainProgram.MyTag2[2], 16	Vom Programm MainProgram das Tag MyTag2 (SINT[20]) ab Index 2 mit Länge 16 (Bereich 2..18)
Byte-Array	Program:MainProgram.MyTag3[4].MyField1[1], 5	Vom Programm MainProgram das Tag MyTag3 (Struktur[5]) bei Index 4 das Feld MyField1 (SInt[10]) ab Index 1 mit Länge 5 (Bereich 1..6)
String	MyStringTag1	Controller-Tag MyStringTag1 (STRING mit max. 82 Characters)
String	MyStringTag2, STRING:20	Controller-Tag MyStringTag2 (benutzerdefinierter String-Typ mit max. 20 Characters)
String-Array	MyStruct.MyStringTag3[2], STRING:20[3]	Controller-Tag MyStruct (Struktur) mit dem Feld MyStringTag3 (benutzerdefinierter String-Typ mit max. 20 Characters) ab Index 2 mit Länge 3 (Bereich 2..5)

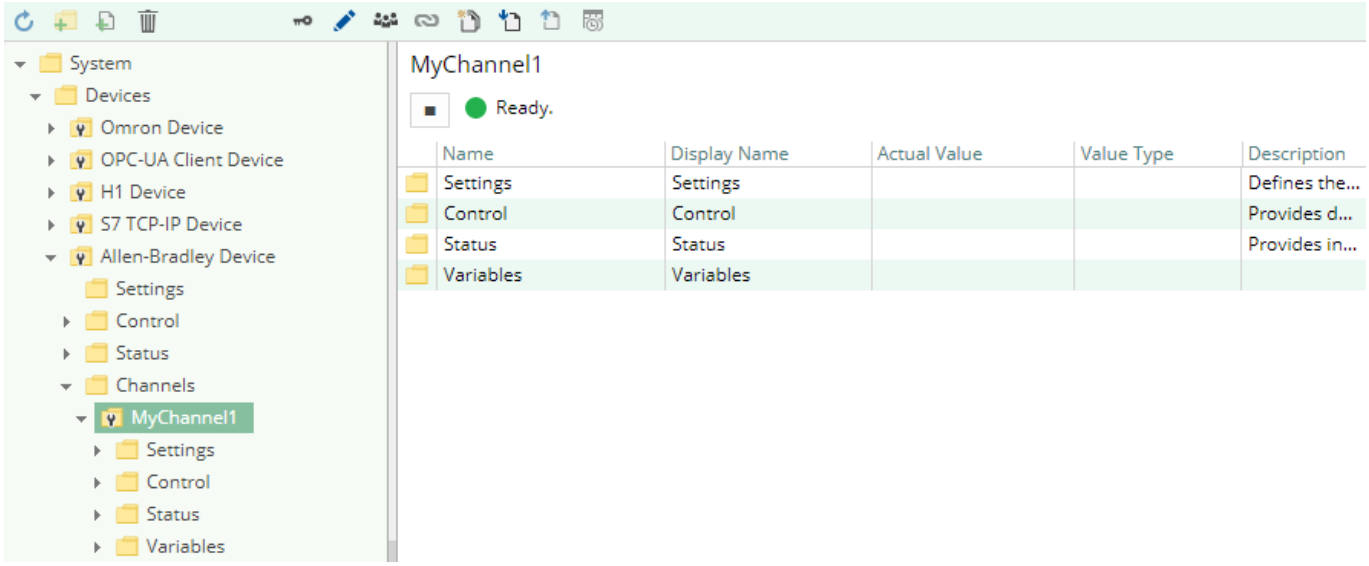
	Name	Display Name	Actual Value	Value Type	Description	Path	Status
	Output	Output	234	Int16		Local:1:O.Data	Good
	Input	Input	0	Int16		Local:1:I.Data	Good
	InputFault	InputFault	0	Int32		Local:1:I.Fault	Good
	MyRealControllerTag	MyRealControllerTag	0	Single		MyRealControllerTag	Good
	Tag1	Tag1	20	Int32		Program:MainProgram.Tag1	Good
	Tag4	Tag4	30,5678	Single		Program:MainProgram.Tag4_REAL	Good
	Array	Array	[10, 20, 30, 99, 200]	Int32-Array		Program:MainProgram.Tag5Array[0], 5	Good
	StructTest1	StructTest1	9999	Int16		Program:MainProgram.MyStructTag1.MyField1	Good
	StructTest2	StructTest2	3,1415	Single		Program:MainProgram.MyStructTag1.MyField2	Good

Fehlerdiagnose

Das Allen-Bradley Device Plugin liefert je nach zu untersuchender Schicht verschiedene Statusinformationen. Generell werden die kanalbasierten Diagnoseinformationen durch den Verbindungsstatus des Channels zur SPS produziert. Die variablenbasierten Diagnoseinformationen werden während des Lese-/Schreibzugriffs auf die verschiedenen Variablen produziert.

Kanal

Um den Status des Allen-Bradley-Kanals zu überwachen und zu diagnostizieren, werfen Sie einen Blick auf das folgende Bild:



Das obige Bild zeigt das Bedienfeld des Allen-Bradley-Kanals, das alle statusrelevanten Informationen anzeigt. Das Bedienfeld aktualisiert automatisch seine Statusinformation, wenn ein neuer Status verfügbar ist.

Statuskreis

Farbe	Bedeutung
	Der Kanal ist gestoppt. Klicken Sie den ►-Button, um ihn zu starten.
	Der Kanal startet oder stoppt gerade oder wartet auf den Verbindungsaufbau.
	Der Kanal ist bereit für Lese-/Schreiboperationen. Sie ihn können durch Klick auf den ■-Button stoppen.
	Der Kanal läuft, aber die Verbindung ist momentan fehlerhaft. Bitte überprüfen Sie den Statustext für weitere Informationen.

Variablen

Um den Status der verschiedenen Variablen zu überwachen und zu diagnostizieren, werfen Sie einen Blick auf die in CoDaBix angezeigte `Status`-Eigenschaft der Spalte. Benutzen Sie den Button „Read actual Value“, um die Werte von der SPS auszulesen und das Ergebnis in den Variablen zu speichern.

Name	Display Name	Actual Value	Value Type	Description	Path	Status
Output	Output		Int16		Local:2:O.Data	Bad: PLCTAG_ERR_NOT_FOUND
Input	Input		Int16		Local:1:I.Data2	Bad: PLCTAG_ERR_BAD_PARAM
InputFault	InputFault		Int32		Local:1:I.	Bad: PLCTAG_ERR_TOO_LARGE
MyRealControllerTag	MyRealControllerTag		Single-Array		MyRealControllerTag, 5	Bad: PLCTAG_ERR_OUT_OF_BOUNDS
Tag2	Tag2		Int32		Program:MainProgram.Tag1	Bad: The operation has timed out.
Tag1	Tag1	20	Int32		Program:MainProgram.Tag1	Good

Häufig vorkommende Fehlertexte:

- **PLCTAG_ERR_NOT_FOUND**: Das Tag wurde in der SPS nicht gefunden. Überprüfen Sie, ob der Name richtig geschrieben wurde.
- **PLCTAG_ERR_BAD_PARAM**: Der Typ oder die Syntax des Tags sind nicht korrekt. Überprüfen Sie den Tagnamen und ob der richtige Value Type verwendet wird.
- **PLCTAG_ERR_TOO_LARGE**: Der gelesene Wert kann nicht in eine Variable von diesem Typ geschrieben, da mehr Daten geliefert wurden, als hineinpassen. Wählen Sie einen Value Type, der dem SPS-Typ entspricht. Falls das Tag eine Struktur ist, geben Sie bitte den Namen des Feldes an, auf das Sie zugreifen möchten.
- **PLCTAG_ERR_OUT_OF_BOUNDS**: Es wurde versucht, auf Arrayindizes zuzugreifen, die außerhalb des Arraybereichs liegen. Überprüfen Sie die angegebene Länge sowie den Offset des Arrays.
- **The operation has timed out.**: Der Zugriff zur SPS hat ein bestimmtes Zeitlimit überschritten, oder die SPS ist nicht erreichbar.

Logdatei

Alle kanalbezogenen Statusinformationen werden auch in die kanalspezifische Logdatei im `[LoggingFolder]` protokolliert. Jede Logdatei wird nach dem Namensschema `Allen-Bradley Device.<ChannelName>.log` benannt.

Der Inhalt einer solchen Logdatei kann wie folgt aussehen:

```
...
2018-04-11 11:32:37.0 +2: [Error] Error (Severity=High): Code=[-1],
Text=[The operation has timed-out.], Details=[]
...
```

Entities

Wie jedes Device Plugin erweitert das Allen-Bradley Device Plugin das CoDaBix Device Modell.

Device

Der Device Typ `AllenBradleyDevice` des Plugins definiert auch den `AllenBradleyDeviceChannel` und erweitert somit die grundlegenden `CodabixDevice` und `CodabixDeviceChannel` Entities. Während das `AllenBradleyDevice` nur eine Konkretisierung des `CodabixDevice` darstellt, erweitert der `AllenBradleyDeviceChannel` den `CodabixDeviceChannel` mit den Allen-Bradley Variable Entities.

Channel

Der Kanal wird von einem Channel Worker behandelt, der eine TCP-Socket-Verbindung zur SPS herstellt.

Der Worker liest standardmäßig keine Werte. Wenn ein Anwender oder Plugin in Codabix einen synchronen Lesevorgang der `Channel`-Variablen anfordert (z.B. mit der „Read actual value“-Funktion in der Codabix Webkonfiguration), liest der Channel Worker diese aus der SPS und schreibt diese in die entsprechenden Codabix-Nodes.

Ähnlich schreibt der Channel Worker auch die Werte in die SPS, wenn ein Client oder Plugin Werte in die `Channel`-Variablen schreibt.

Damit eine Allen-Bradley-Variable regelmäßig gelesen wird, können Sie in der Webkonfiguration bei dem Node „History Options“ auf `Yes` stellen (was eine interne Subscription erstellt), oder Sie können zum Beispiel einen OPC UA Client verwenden, der mit dem OPC UA Server Plugin verbunden ist und damit eine Subscription für die Allen-Bradley Variablennodes erstellen. In diesen Fällen liest der Channel Worker in regelmäßigen Intervallen die Variablen von der SPS und schreibt den neuen Wert nach einer Wertänderung automatisch in dden entsprechenden Codabix-Node.

Ordner & Dateien

Ordner

Name	Pfad	Zweck / Verwendung
AssemblyFolder	`<CodabixInstallDir>/plugins/AllenBradleyDevicePlugin/`	Beinhaltet die Plugin-Assemblydatei.
ConfigFolder	`<CodabixProjectDir>/plugins/AllenBradleyDevicePlugin/`	Beinhaltet die Plugin-Konfigurationsdatei.
LoggingFolder	`<CodabixProjectDir>/log/`	Beinhaltet die Plugin-Logdateien.

Dateien

Typ	Pfad	Zweck / Verwendung
Assembly	`[AssemblyFolder]/CoDaBix.AllenBradleyDevicePlugin.dll`	Die Plugin-Assembly Datei.
Logging	`[LoggingFolder]/Allen-Bradley Device.<ChannelName>.log`	Die Logdatei.

Versionsinformation

Dieses Dokument

Datum	`2019-11-11`
Version	`1.0`

Plugin

Name	Allen-Bradley Device Plugin
Node	/System/Devices/Allen-Bradley Device
Version	1.0.0

Assembly

Name	CoDaBix.AllenBradleyDevicePlugin.dll
Datum	2019-11-11
Version	1.0.0.0

From:
https://www.codabix.de/ - **CoDaBix®**

Permanent link:
https://www.codabix.de/de/plugins/device/allenbradleydeviceplugin

Last update: **2021/07/30 13:39**