DOKUMENTATION

PLCcom für Modbus
Anhang E

PLCcom
by Indi.An Germany

Technische Produktdokumentation

CANopen Guide

PLCcom für Modbus · Java

Dokument-ID	IND-MODBUS-JAVA-CANOPEN-DE
Dokument-Version	1.0
SDK-Version	9.x
Sprache	Deutsch
Plattform	Java
Herausgeber	Indi.An GmbH

Indi.An GmbH · Flughafenallee 3 · D-28199 Bremen
support@indi-an.com · Tel +49 421 989703-30

PLCcom für Modbus Java — Anhang E: CANopen Guide

Dokument-ID: IND-MODBUS-JAVA-CANOPEN-DE
Dokumentversion: 1.0
SDK-Version: 9.x
Sprache: Deutsch
Plattform: Java

Inhaltsverzeichnis

1. Über dieses Dokument
2. Einordnung
3. CANopen-Grundbegriffe
4. Object Dictionary
5. Indexbereiche und Gerätedokumentation
6. SDO, PDO und der Bezug zu Modbus
7. Zugriff über Modbus
8. Parametrierung in PLCcom
9. Lesen eines Objekts
10. Schreiben eines Objekts
11. Datentypen, Registerlänge und Byteorder
12. Diagnose und typische Fehler
13. Grenzen und Geräteabhängigkeit
14. Beispiele
15. Weiterführende Hinweise
16. Fragen und Support

1. Über dieses Dokument

Dieser Guide beschreibt den CANopen-style Object Dictionary Access über Modbus in PLCcom für Modbus Java.

Gemeint sind Geräte, Gateways oder Antriebe, die intern mit einem CANopen-ähnlichen Objektmodell arbeiten und diese Objekte zusätzlich über Modbus zugänglich machen.

PLCcom für Modbus wird dadurch nicht zu einem vollständigen CANopen-Stack für den CAN-Bus.
Die Bibliothek bleibt eine Modbus-Bibliothek. Sie unterstützt jedoch Zugriffsmuster, bei denen der Anwender mit Index, Subindex, Datentyp und Gerätehandbuch arbeitet, wie man es aus CANopen kennt.

Das Dokument richtet sich an Integratoren, die ein solches Gerät über Modbus parametrieren, diagnostizieren oder auslesen möchten.

2. Einordnung

CANopen ist ein Kommunikations- und Geräteprofilkonzept aus der Automatisierungstechnik. Der zentrale Gedanke ist, dass ein Gerät seine Kommunikationsparameter, Konfigurationswerte und Nutzdaten in einem sogenannten Object Dictionary bereitstellt.

Ein klassisches CANopen-Gerät wird über CANopen-Dienste angesprochen. In der Praxis gibt es aber Geräte, die dieses Objektmodell auch über andere Kommunikationswege anbieten. Dazu gehören zum Beispiel Umrichter, Gateways oder Steuerungskomponenten, die CANopen-nahe Parameter über Modbus-Register oder eine Modbus-Mailbox erreichbar machen.

Für den Integrator bedeutet das:

- Modbus bleibt der Transportweg.
- Die logische Adresse ist häufig nicht nur eine Registeradresse, sondern ein CANopen-Objekt.
- Der Zugriff wird durch Index, Subindex, Datentyp, Zugriffstyp und Gerätehandbuch bestimmt.
- Die genaue Modbus-Abbildung ist herstellerspezifisch.

PLCcom hilft an dieser Stelle, solche Zugriffe aus der Anwendung heraus sauber zu strukturieren. Die Bibliothek kann aber nicht erraten, wie ein bestimmter Hersteller sein Object Dictionary auf Modbus abbildet.

3. CANopen-Grundbegriffe

Die folgenden Begriffe tauchen in Gerätehandbüchern häufig auf. Sie sind wichtig, um die Modbus-Abbildung des Geräts richtig zu lesen.

Begriff	Bedeutung
Node-ID	Knotenadresse eines CANopen-Geräts. Bei einem reinen Modbus-Zugriff kann sie durch die Slave-ID ersetzt, fest vorgegeben oder Teil einer Gateway-Struktur sein.
Object Dictionary	Objektverzeichnis des Geräts. Es enthält Kommunikationsparameter, Diagnosewerte, Konfigurationswerte und Prozessdaten.
Index	16-Bit-Adresse eines Objekts im Object Dictionary, häufig hexadezimal geschrieben, z. B. `0x1017`.
Subindex	Untereintrag eines Objekts. Viele Objekte sind Arrays oder Records und benötigen deshalb zusätzlich zum Index einen Subindex.
SDO	Service Data Object. Dient in CANopen zum gezielten Lesen und Schreiben einzelner Object-Dictionary-Einträge.
PDO	Process Data Object. Dient in CANopen für zyklische oder ereignisgesteuerte Prozessdaten.
EDS	Electronic Data Sheet. Beschreibungsdatei eines CANopen-Geräts mit Objektverzeichnis und Eigenschaften.
DCF	Device Configuration File. Konfigurierte Gerätebeschreibung auf Basis einer EDS-Datei.

Wenn ein Hersteller im Modbus-Handbuch von CANopen-Objekten, SDO-Zugriff, Index/Subindex oder Object Dictionary spricht, ist in der Regel genau dieser Zusammenhang gemeint.

4. Object Dictionary

Das Object Dictionary ist die zentrale Tabelle eines CANopen-Geräts. Jeder Eintrag beschreibt einen Wert oder eine Struktur, die gelesen, geschrieben oder zur Diagnose verwendet werden kann.

Ein Eintrag besteht typischerweise aus:

Feld	Beschreibung
Index	16-Bit-Adresse des Objekts.
Subindex	Untereintrag innerhalb des Objekts.
Name	Lesbarer Objektname aus Geräteprofil oder Herstellerdokumentation.
Objekttyp	Variable, Array, Record oder herstellerspezifische Struktur.
Datentyp	Zum Beispiel `UNSIGNED8`, `UNSIGNED16`, `UNSIGNED32`, `INTEGER32`, `REAL32` oder ein String-Typ.
Zugriff	Zum Beispiel `read only`, `write only` oder `read/write`.
Bedeutung	Fachliche Interpretation, Einheit, Skalierung und zulässiger Wertebereich.

Beispielhafte Schreibweise:

```
Index:     0x2000
Subindex:  0x01
Datentyp:  UNSIGNED16
Zugriff:   read/write
Einheit:   herstellerspezifisch
```

Wichtig ist: Index und Subindex beschreiben noch nicht automatisch eine Modbus-Adresse.

Erst die Gerätedokumentation legt fest, ob und wie dieses Objekt über Modbus erreichbar ist. Manche Geräte besitzen eine direkte Registertabelle. Andere Geräte verwenden einen kleinen Befehlsbereich, in den Index, Subindex und Zugriffstyp geschrieben werden. Wieder andere Geräte bilden nur ausgewählte Objekte auf Modbus ab.

5. Indexbereiche und Gerätedokumentation

CANopen verwendet etablierte Indexbereiche. Sie helfen bei der Orientierung, ersetzen aber nicht das Gerätehandbuch.

Bereich	Typische Bedeutung
`0x0001` bis `0x025F`	Datentypen und zusammengesetzte Typdefinitionen.
`0x1000` bis `0x1FFF`	Kommunikationsprofilbereich, z. B. Gerätetyp, Error Register, Heartbeat oder SDO/PDO-Parameter.
`0x2000` bis `0x5FFF`	Herstellerbereich. Viele Geräte legen hier eigene Parameter, Diagnosewerte und Betriebsarten ab.
`0x6000` bis `0x9FFF`	Standardisierte Geräteprofile, z. B. I/O-, Antriebs- oder Sensorprofile.

In vielen Integrationsprojekten liegt der wichtigste Teil im Herstellerbereich. Genau dort unterscheiden sich Geräte besonders stark.

Deshalb sollte vor der Implementierung geklärt werden:

1. Unterstützt das Gerät überhaupt einen CANopen-style Zugriff über Modbus?
2. Welche Objekte sind über Modbus erreichbar?
3. Wird direkt auf Registeradressen gemappt oder gibt es eine Request-/Response-Struktur?
4. Welche Datentypen und Längen werden erwartet?
5. Welche Byteorder gilt für Werte mit mehr als 16 Bit?
6. Welche Zustände des Geräts erlauben Schreibzugriffe?
7. Wie werden Fehler gemeldet?

Ein Objekt kann in der CANopen-Beschreibung vorhanden sein und trotzdem nicht über Modbus erreichbar sein.
Diese Unterscheidung ist wichtig, weil sie viele vermeintliche Kommunikationsfehler erklärt.

6. SDO, PDO und der Bezug zu Modbus

In CANopen wird der gezielte Zugriff auf Object-Dictionary-Einträge üblicherweise über SDOs beschrieben.

Ein SDO-Lesezugriff wird in CANopen-Terminologie als Upload bezeichnet.
Ein SDO-Schreibzugriff wird als Download bezeichnet.

Das klingt für neue Anwender oft ungewohnt:

Anwendersicht	CANopen-Begriff
Wert aus dem Gerät lesen	SDO Upload
Wert in das Gerät schreiben	SDO Download

PDOs sind dagegen für Prozessdaten gedacht. Ein PDO kann mehrere Object-Dictionary-Werte enthalten und wird typischerweise zyklisch, ereignisgesteuert oder synchron übertragen. PDOs sind für schnelle Prozessdaten geeignet, aber weniger für einzelne Parametrier- oder Diagnosezugriffe.

Bei PLCcom für Modbus geht es in diesem Guide nicht darum, einen CANopen-Bus mit SDO- und PDO-Telegrammen aufzubauen. Es geht darum, Modbus-Geräte zu bedienen, deren Hersteller eine CANopen-nahe Objektadressierung über Modbus bereitstellt.

Wenn ein Handbuch also von “SDO access via Modbus”, “Object Dictionary access”, “Index/Subindex” oder ähnlichen Begriffen spricht, kann PLCcom diesen Zugriff aus der Modbus-Anwendung heraus abbilden, sofern die benötigten Modbus-Details dokumentiert sind.

7. Zugriff über Modbus

Die Abbildung von CANopen-style Objektzugriffen auf Modbus ist nicht einheitlich normiert. Sie muss immer aus der Dokumentation des Zielgeräts übernommen werden.

In der Praxis treten vor allem diese Varianten auf:

Variante	Beschreibung
Direkte Registerabbildung	Ein Objekt oder Subobjekt ist auf eine feste Modbus-Adresse gelegt. Der Anwender liest oder schreibt diese Adresse wie ein normales Register.
Mailbox-/Command-Bereich	Die Anwendung schreibt Index, Subindex, Zugriffstyp und Nutzdaten in definierte Register. Danach wird ein Status- oder Ergebnisbereich gelesen.
Gateway-Abbildung	Ein Modbus-Gerät wirkt als Gateway zu CANopen-Knoten. Die Modbus-Slave-ID, eine Node-ID oder ein Registerfeld bestimmt den Zielknoten.
Herstellerfunktion	Das Gerät nutzt eine dokumentierte, aber proprietäre Abbildung. Sie kann sich von Hersteller zu Hersteller deutlich unterscheiden.

Für PLCcom ist entscheidend, welche Modbus-Operation am Ende ausgeführt werden muss:

- Holding Register lesen oder schreiben
- Input Register lesen
- mehrere Register als zusammenhängenden Block übertragen
- einen Request-Bereich beschreiben und anschließend einen Response-Bereich abfragen
- Fehlerstatus oder Quality auswerten

Die CANopen-Begriffe beschreiben das fachliche Ziel.
Die Modbus-Beschreibung des Herstellers beschreibt den tatsächlich auszuführenden Telegrammverkehr.

8. Parametrierung in PLCcom

Ein CANopen-style Zugriff benötigt dieselben Grunddaten wie jeder andere Modbus-Zugriff, ergänzt um die Objektinformationen aus dem Gerätehandbuch.

Typische Parameter sind:

Parameter	Bedeutung
Verbindungsart	Zum Beispiel TCP, RTU oder eine andere vom Gerät unterstützte Modbus-Variante.
Slave-ID	Modbus-Adresse des Zielgeräts oder Gateways.
Index	CANopen-Objektindex, z. B. `0x2000`.
Subindex	Unterobjekt, z. B. `0x01`.
Datentyp	Erwartete Interpretation des Nutzwertes.
Zugriff	Lesen oder Schreiben.
Registerbereich	Modbus-Adresse oder Bereich, über den der Hersteller den Zugriff abbildet.
Byteorder	Byte- und Word-Reihenfolge für Werte größer als 16 Bit.
Skalierung	Faktor, Einheit oder Umrechnung aus dem Gerätehandbuch.

Bei einfachen Geräten genügt möglicherweise eine direkte Registeradresse.
Bei komplexeren Geräten muss zuerst ein Kommando geschrieben und danach ein Ergebnis gelesen werden.

Die Anwendung sollte diese Parameter nicht im Code erraten. Sinnvoll ist eine kleine, gut lesbare Zuordnungstabelle im Projekt, die aus dem Gerätehandbuch abgeleitet wird.

Beispiel:

```
Parameter:       Motor current limit
Index:           0x2301
Subindex:        0x02
Datentyp:        UNSIGNED16
Einheit:         0.1 A
Zugriff:         read/write
Modbus-Abbild.:  laut Gerätehandbuch
```

9. Lesen eines Objekts

Ein Lesezugriff sollte nicht nur den Nutzwert ausgeben, sondern immer auch die Qualität und mögliche Gerätefehler berücksichtigen.

Ein typischer Ablauf ist:

1. Verbindung zum Gerät herstellen.
2. Objekt aus der Dokumentation auswählen.
3. Index, Subindex, Datentyp und Modbus-Abbildung prüfen.
4. PLCcom-Request aufbauen.
5. Request ausführen.
6. Ergebnisqualität prüfen.
7. Rohwert in den fachlichen Wert umrechnen.
8. Telegramm- und Diagnoseausgabe bei Bedarf protokollieren.

Bei Werten größer als 16 Bit muss zusätzlich die Registerreihenfolge stimmen.
Ein falscher Wert bedeutet nicht automatisch, dass der Zugriff fehlgeschlagen ist. Häufig ist der Request technisch erfolgreich, aber Datentyp, Skalierung oder Byteorder passen noch nicht zur Gerätebeschreibung.

Gerade bei herstellerspezifischen Objekten lohnt es sich, zunächst mit einem gut dokumentierten, ungefährlichen Read-only-Objekt zu starten.

10. Schreiben eines Objekts

Schreibzugriffe sollten vorsichtiger behandelt werden als Lesezugriffe. Sie verändern Geräteparameter und können je nach Gerät unmittelbar Einfluss auf den Betrieb haben.

Vor einem Schreibzugriff sollte geprüft werden:

- Ist das Objekt laut Dokumentation schreibbar?
- Ist der aktuelle Gerätezustand für diesen Schreibzugriff erlaubt?
- Muss der Wert skaliert werden?
- Muss der Wert nach dem Schreiben gespeichert werden?
- Wird ein Neustart oder ein erneutes Aktivieren des Parameters benötigt?
- Gibt es Grenzwerte oder reservierte Werte?

Viele Geräte akzeptieren Schreibzugriffe nur in bestimmten Zuständen. Bei Antrieben kann zum Beispiel ein Parameter im Betriebszustand gesperrt sein, obwohl er grundsätzlich als `read/write` dokumentiert ist.

Ein erfolgreicher Modbus-Schreibrequest bedeutet außerdem nicht immer, dass der Parameter fachlich übernommen wurde. Manche Geräte melden eine spätere Ablehnung über einen Statusbereich, ein Error Register oder eine herstellerspezifische Diagnose.

Für produktive Anwendungen ist deshalb empfehlenswert:

1. Wert schreiben.
2. Ergebnisqualität prüfen.
3. Gerätedokumentierten Status prüfen.
4. Wert erneut lesen.
5. Den übernommenen Wert mit dem Sollwert vergleichen.

11. Datentypen, Registerlänge und Byteorder

Modbus-Register sind 16 Bit breit. Viele CANopen-Datentypen sind jedoch 8, 16, 32 oder 64 Bit breit. Dadurch entsteht eine Übersetzungsschicht zwischen Object Dictionary und Modbus-Registern.

Typische Fälle:

Datentyp	Typische Modbus-Länge	Hinweise
`UNSIGNED8` / `INTEGER8`	Teil eines 16-Bit-Registers	Gerätedokumentation muss beschreiben, welches Byte verwendet wird.
`UNSIGNED16` / `INTEGER16`	1 Register	Der einfachste Fall.
`UNSIGNED32` / `INTEGER32` / `REAL32`	2 Register	Word-Reihenfolge und Byteorder beachten.
`UNSIGNED64` / `INTEGER64` / `REAL64`	4 Register	Nur verwenden, wenn das Gerät diesen Zugriff ausdrücklich unterstützt.
String oder Array	mehrere Register	Länge, Terminierung und Zeichensatz sind herstellerspezifisch.

Die Byteorder ist besonders wichtig, sobald mehr als ein Register beteiligt ist.

Ein Gerät kann zum Beispiel einen 32-Bit-Wert in zwei Registern so übertragen:

```
Register 0: High Word
Register 1: Low Word
```

Ein anderes Gerät kann dieselben Daten umgekehrt übertragen:

```
Register 0: Low Word
Register 1: High Word
```

Zusätzlich kann innerhalb eines Words die Byte-Reihenfolge relevant sein. Deshalb sollte die Byteorder nicht nach Gefühl eingestellt werden. Sie muss aus der Gerätedokumentation oder durch einen bekannten Testwert bestimmt werden.

Ein guter Testwert ist ein Wert, dessen Bytefolge eindeutig erkennbar ist, zum Beispiel ein dokumentierter Versionswert, ein fester Skalierungsfaktor oder ein Herstellerparameter mit bekanntem Inhalt.

12. Diagnose und typische Fehler

CANopen-style Zugriffe über Modbus schlagen häufig nicht wegen der TCP- oder RTU-Verbindung fehl, sondern wegen einer falschen fachlichen Abbildung.

Die folgende Tabelle hilft bei der ersten Einordnung:

Symptom	Mögliche Ursache
Modbus-Exception "Illegal Function"	Das Gerät unterstützt die verwendete Modbus-Funktion nicht oder der Zugriff muss über eine andere Abbildung erfolgen.
Modbus-Exception "Illegal Data Address"	Registeradresse, Mailbox-Bereich oder Objektabbildung passt nicht zum Gerät.
Timeout	Falscher Port, falsche Slave-ID, Gateway beschäftigt oder Wartezeit nach Verbindungsaufbau zu kurz.
Gute Quality, aber falscher Wert	Datentyp, Skalierung, Registermode oder Byteorder passt nicht.
Schreibrequest erfolgreich, Wert bleibt unverändert	Objekt ist nicht schreibbar, Gerät ist im falschen Zustand oder Parameter muss separat gespeichert werden.
Gerätefehler im Statusbereich	Das Gerät hat den Object-Dictionary-Zugriff fachlich abgelehnt.
Zugriff funktioniert mit einem Objekt, aber nicht mit einem anderen	Das zweite Objekt ist über Modbus nicht freigegeben oder nur in bestimmten Zuständen erlaubt.

Bei der Fehlersuche sollten Telegramm-Logging und Gerätehandbuch gemeinsam verwendet werden.

Das Telegramm zeigt, was tatsächlich über Modbus übertragen wurde.
Das Gerätehandbuch erklärt, wie der Wert fachlich zu interpretieren ist.

13. Grenzen und Geräteabhängigkeit

CANopen-style Zugriff über Modbus ist kein universeller Modbus-Standard. Er ist ein Geräte- oder Gateway-Feature.

Deshalb gelten einige wichtige Grenzen:

- Ein normales Modbus-Gerät versteht nicht automatisch CANopen-Objekte.
- Ein CANopen-Objekt ist nicht automatisch über Modbus erreichbar.
- Ein identischer Index kann bei zwei Herstellern unterschiedlich abgebildet sein, insbesondere im Herstellerbereich.
- Schreibzugriffe können vom Gerätezustand abhängen.
- Fehlercodes können als Modbus-Exception, Statusregister oder herstellerspezifischer Rückgabewert erscheinen.
- EDS- oder DCF-Dateien helfen beim Verständnis des Object Dictionary, ersetzen aber nicht die Modbus-Mapping-Dokumentation.

Für eine stabile Integration sollte die Anwendung deshalb immer defensiv arbeiten:

1. Verbindungsfehler und Modbus-Exceptions sauber behandeln.
2. Antwortqualität auswerten.
3. Gerätefehler getrennt von Transportfehlern protokollieren.
4. Byteorder und Skalierung explizit dokumentieren.
5. Schreibzugriffe durch Rücklesen validieren.

14. Beispiele

Die Java-Beispiele enthalten ein CANopen-Beispiel.

→ **https://github.com/Indi-An/PLCcom-modbus-example-java**

Das Beispiel zeigt den grundsätzlichen Aufbau eines CANopen-style Objektzugriffs über Modbus. Für produktive Geräte müssen Index, Subindex, Datentyp, Byteorder, Skalierung und Kommunikationsparameter anhand der Gerätedokumentation angepasst werden.

Die Beispiele sollen den Einstieg erleichtern, ersetzen aber kein herstellerspezifisches Integrationshandbuch.

15. Weiterführende Hinweise

Für das Verständnis der Begriffe Object Dictionary, SDO, PDO, EDS und DCF sind CANopen-Grundlagen hilfreich. Maßgeblich sind die Spezifikationen und Profile von CAN in Automation, insbesondere die CANopen Application Layer and Communication Profile-Familie.

Hilfreiche Suchbegriffe in Gerätehandbüchern sind:

- `Object Dictionary`
- `Index`
- `Subindex`
- `SDO`
- `SDO upload`
- `SDO download`
- `EDS`
- `DCF`
- `CANopen over Modbus`
- `Object dictionary access via Modbus`
- `Modbus mailbox`

Wenn ein Gerät diese Begriffe nicht verwendet, aber nur eine klassische Modbus-Registertabelle anbietet, sollte der Zugriff als normaler Modbus-Registerzugriff umgesetzt werden.

16. Praktischer Mapping-Ablauf

Bei CANopen-style Zugriffen über Modbus besteht die entscheidende Arbeit darin, die Gerätedokumentation auf konkrete Modbus-Requests abzubilden. Ein sinnvoller Ablauf ist:

1. Objekt über Index und Subindex identifizieren.
2. Prüfen, ob das Objekt wirklich über die Modbus-Schnittstelle des Geräts erreichbar ist.
3. Klären, ob das Gerät direkte Register oder einen Mailbox-/Command-Bereich verwendet.
4. Zugriffsrichtung bestimmen: lesen, schreiben oder lesen/schreiben.
5. Datentyp, Länge, Vorzeichen und Skalierung bestimmen.
6. Byteorder und Registermodus bestimmen.
7. Einen Lesezugriff ausführen und den Rohwert mit einem bekannten Gerätewert vergleichen.
8. Schreibzugriffe erst danach implementieren.

16.1 Direkte Abbildung oder Mailbox-Abbildung

Einige Geräte bilden Object-Dictionary-Einträge direkt in Modbus-Register ab. Dann liest oder schreibt die Anwendung normale Register; die CANopen-Begriffe dienen vor allem dazu, das Handbuch richtig zu verstehen.

Andere Geräte verwenden einen Mailbox-Bereich. Die Anwendung schreibt Index, Subindex, Kommando und Nutzdaten in definierte Register und fragt danach einen Status- oder Antwortbereich ab. Das ist komplexer, weil Busy-Zustände, Antwortcodes und gerätespezifische Timeouts behandelt werden müssen.

Abbildungsart	Schwerpunkt in der Anwendung
direkte Registerabbildung	korrekte Adresse, Datentyp, Byteorder und Skalierung
Mailbox-Abbildung	Kommandosequenz, Statuspolling, Antwortauswertung und Timeout-Behandlung

16.2 Schreibzugriff und Gerätezustand

Viele CANopen-style Objekte können nur in bestimmten Gerätezuständen geschrieben werden. Antriebe lehnen Parameteränderungen zum Beispiel häufig ab, wenn sie freigegeben sind oder ein Fehler aktiv ist. Ein erfolgreicher Modbus-Transport bedeutet deshalb nicht automatisch, dass der fachliche Objektzugriff akzeptiert wurde. Gerätespezifische Statusinformationen sollten immer ausgewertet und kritische Parameter zurückgelesen werden.

16.3 Dokumentation beim Code halten

Für jedes von der Anwendung verwendete Objekt sollte eine kleine Mapping-Notiz im Codeumfeld oder in der Projektdokumentation bleiben:

- Objektindex und Subindex
- Objektname aus dem Handbuch
- Modbus-Adresse oder Mailbox-Kommando
- Datentyp und Länge
- Zugriffsrechte
- Skalierung und Einheit
- Byteorder
- erlaubte Gerätezustände
- erwartete Fehler- oder Statuscodes

So hängt spätere Wartung nicht vom Gedächtnis oder von einzelnen Inbetriebnahmenotizen ab.

17. Fragen und Support

Für technische Fragen, Lizenzthemen und Support wenden Sie sich an Indi.An GmbH bzw. an die in Ihrem Lizenz- oder Wartungsvertrag genannten Ansprechpartner.