Erste Schritte mit dem PLCcom SDK V.11

Indi.An Systems GmbH

Flughafenallee 3
28199 Bremen
Deutschland
support@indi-systems.de

Tel + 49 421-989703-30
Fax + 49 421-989703-39

Inhaltsverzeichnis

Über dieses Dokument? ... 3
Was bietet Ihnen PLCcom für S7? ... 4
Welche Programmierplattformen unterstützt PLCcom? ... 5
Welche System-Komponenten sind Voraussetzung für den Betrieb von PLCcom? ... 6
Wichtige Hinweise zur Benutzung von PLCcom unter Android ... 6
Übergeben des Lizenzschlüssels ... 7
Erste Programmier-Schritte mit PLCcom ... 8
Syntax: PLCcom-Device-Objekt erstellen und initiieren ... 8
Syntax: Erstellen eines PLCcom-Requests z.B. Lesen von Werten ... 9
Syntax: Übergabe des Requests an das PLCcom-Device und Empfangen des Results ... 10
Syntax: Auswerten des Requests ... 10
Funktionsübersicht ... 12
Einfaches Lesen von Daten aus der SPS ... 12
Einfaches Schreiben von Daten in die SPS ... 14
Optimiertes Lesen und Schreiben von Daten ... 16
Optimierungsmöglichkeiten ... 17
SPS starten ... 21
SPS stoppen ... 22
SPS Zeit lesen ... 23
SPS Zeit setzen ... 24
SPS Basic-Infos auslesen ... 25
CPU-Modus auslesen ... 26
LED-Status auslesen ... 27
Systemzustandslisten SZL lesen ... 28
Diagnosedaten lesen ... 29
PLCCom-DataServer (nur in der Expert-Version) ... 30
Passwort senden (nur in der Expert-Version) ... 34
Objektliste aus SPS abrufen (nur in der Expert-Version) ... 35
Länge eines Objektes abrufen (nur in der Expert-Version) ... 36
Backup eines Objektes (nur in der Expert-Version) ... 37
Restore eines Objektes (nur in der Expert-Version) ... 39
Löschen eines Objektes in der SPS (nur in der Expert-Version) ... 42

Indi.An Systems GmbH
Flughafenallee 3
28199 Bremen
Deutschland
support@indi-systems.de
Tel + 49 421-989703-30
Fax + 49 421-989703-39

Haben Sie Fragen? 43

Über dieses Dokument?

Das vorliegende Dokument soll Ihnen einen ersten Überblick über die bereitgestellten Funktionalitäten liefern. Es handelt sich nicht um eine komplette Dokumentation, sondern wird bereitgestellt um Ihnen einen ersten Einstieg zu ermöglichen. Weitergehende Informationen entnehmen Sie bitte

- den Code-Beispielen im Auslieferungspaket,
- den Code-Beispielen und der Onlinehilfe auf unserer Website http://www.plccom.net/code-examples/plccom/s7.html
- sowie der jeweiligen Onlinehilfe im Auslieferungspaket (index.html)

Was bietet Ihnen PLCcom für S7?

Die Library PLCcom ist eine speziell für Java / .NET-Softwareentwickler bereitgestellte hoch optimierte und moderne Komponente, um Softwareentwicklern komfortabel Zugriff auf eine Siemens S7-SPS zur Verfügung zu stellen, um Daten auszulesen oder zu schreiben.

Bei den Librarys handelt es sich je nach Version um 100% Java oder .Net-Dateien. Der Treiber kann direkt als Verweis eingebunden werden, API-Aufrufe sind nicht notwendig. Es ist problemlos möglich, die Komponente in 32-oder 64 Bit-Umgebungen sowie plattformübergreifend einzusetzen.

Die internen Routinen sind auf High-Performance-Zugriffe optimiert.

Mit im Lieferumfang enthalten sind umfangreiche Code-Beispiele, die die kinderleichte Anbindung der SPS-Steuerung an Ihre Applikation verdeutlichen und auch in Ihren Projekten genutzt werden können.

PLCcom ist in aktueller Version kompatibel zu S7-Steuerungen (200er, 300er, 400er, 1200er, 1500er Baureihe, SoftSPS WinAC RTX sowie Logo! 0BA7 / 0BA8/ 0BA0) sowie CPUs weiterer Hersteller (z. B. VIPA 100V/200V/300V/300S, etc.)

Weitere Merkmale der PLCcom für S7-Library:

- Lese- und Schreiboperationen von folgenden Datentypen
  - Rohdaten (Byte oder Bool)
  - Bit
  - Byte
  - Word
  - DWord
  - Float
  - INT
  - DINT
  - 16Bit BCD
  - DATE_AND_TIME
  - String
  - S7String
- Lesen von Eingängen, Ausgängen, Datenbausteinen, Merkern, Timern, Countern
- Mehrere Operationen in einem Funktionsaufruf möglich, jede Operation gibt einen detaillierten Rückgabewert wieder.
- Gleichzeitiger Zugriff auf verschiedene CPUs möglich
- Implementierter High-Performance-Zugriff, es werden die benötigten Zugriffe auf die SPS auf das absolut notwendige Minimum reduziert.
- Interne Hilfs- und Umwandlungsfunktionen für das einfache Lesen und Schreiben bestimmter SPS-Datentypen
- Bereitstellung des PLCcom-Dataservers für zyklische Lesezugriffe mit eventgesteuerter Benachrichtigung bei Werteänderung
- Starten und Stoppen der CPU

Indi.An Systems GmbH
Flughafenallee 3
28199 Bremen
Deutschland
support@indi-systems.de
Tel + 49 421-989703-30
Fax + 49 421-989703-39

- Auslesen von Seriennummer und Firmwareversion
- Auslesen der Schlüsselschalterstellung
- Auslesen von LED-Infos
- Lesen und setzen der SPS-Zeit
- Lesen der Systemzustandsliste
- Lesen der Blockliste
- Lesen von Blocklängen
- Backup von Blöcken
- Restore von Blöcken
- Restore von Blöcken unter geänderter Blocknummer
- Löschen von Blöcken
- Ausgabe von Blockdetails wie Code, Erstellungssprachen, Author uvm.
- Senden des Verbindungspasswortes
- Auslesen der Diagnosedaten
- AutoConnect
- asynchones Verbinden
- u.v.m.

Welche Programmierplattformen unterstützt PLCcom?

PLCcom für S7 wird in drei Versionen zur Verfügung gestellt:

1. .Net Version
   Die .Net-Version unterstützt die klassische .Net Framework-Programmierung.
   Des Weiteren enthält das Auslieferungspaket eine Version für **.Net-Standard Version 2.1**
   Diese Komponente kann zur Entwicklung von **.Net-Core- , Xamarin- , UWP- oder Unity-Applikationen** eingesetzt werden.

2. Windows CE-Version
   Es wird die Entwicklung von Applikationen unter Windows CE ab Version 5 unterstützt.

3. Java-Version
   Die Java-Version bietet Entwicklern eine Java-Komponente zur Java-Applikationsentwicklung z.B. mit Eclipse- oder Netbeans.
   Seit Version 9 wird zusätzlich auch die Entwicklung von **Android-Apps** unterstützt.

Indi.An Systems GmbH
Flughafenallee 3
28199 Bremen
Deutschland
support@indi-systems.de
Tel + 49 421-989703-30
Fax + 49 421-989703-39

Welche System-Komponenten sind Voraussetzung für den Betrieb von PLCcom?

Für den Betrieb von PLCcom sind folgende System-Komponenten Voraussetzung:

- Microsoft .NET Framework 3.5 oder neuer (.Net Version)
- Microsoft .NET Core 3.1 oder neuer (.Net Version)
- Microsoft .NET 5.0 oder neuer (.Net Version)
- Xamarin.iOS Version 12.16 oder neuer (.Net Version)
- Xamarin.Mac Version 5.16 oder neuer (.Net Version)
- Xamarin.Android Version 10.0 oder neuer (.Net Version)
- Mono Version 6.0 oder neuer (.Net Version)
- Microsoft .NET Compact Framework 3.5 (CE Version)
- Java JRE8 / OpenJDK8 oder neuer (Java Version)
- Android API 21 oder neuer (Java Version)
- Externe Komponente Jssc zum Bereitstellen von Serialports unter Java (Java Version, aber nicht für Android siehe Kapitel „Wichtige Hinweise zur Benutzung von PLCcom unter Android“)

Um die beigefügten Programmbeispiele ausführen zu können, benötigen Sie folgende Programmierwerkzeuge:

- Visual Studio 2019 oder neuer (.Net-Version)
- Visual Studio 2008 (CE Version)
- Netbeans 8 oder 4.5 oder neuer (Java Version)
- Eclipse in Version Oxigen oder neuer (Java Version)

Wichtige Hinweise zur Benutzung von PLCcom unter Android

Aufgrund des Fehlens von seriellen Schnittstellen, ist die Benutzung folgender Objekte unter Android nicht möglich:

PLCCom.MPI_Device
PLCCom.PPI_Device

Der Versuch eine Instanz der oben genannten Klassen zu erstellen, führt zu einem Error des Types **java.lang.NoClassDefFoundError**

Des Weiteren muss für die Benutzung des Objektes **PLCCom.FileSystemConnector** abweichend zu den oben genannten Angaben mindestens die Android API Version 26 oder neuer benutzt werden.

Aufgrund der allgemeinen Systemgegebenheiten unter Android wird davon ausgegangen, dass die Benutzung der oben genannten Objekte unter Android keine Rolle spielt und die Einschränkungen zu vernachlässigen sind.

Übergeben des Lizenzschlüssels

Vor der Benutzung der PLCcom-Library müssen Sie im Besitz eines Lizenzschlüssels für die aktuelle Version sein. Diesen Lizenzschlüssel können Sie entweder als zeitlich befristeten Trial-Schlüssel beziehen, oder im Vorfeld einen Lizenzschlüssel käuflich erworben haben.

Dieser Lizenzschlüssels muss wie folgt an die Library übergeben werden:

CSharp

```
authentication.User = "your user name";
authentication.Serial = "your user serial key";
```

Visual Basic

```
authentication.User = "your user name"
authentication.Serial = "your user serial key"
```

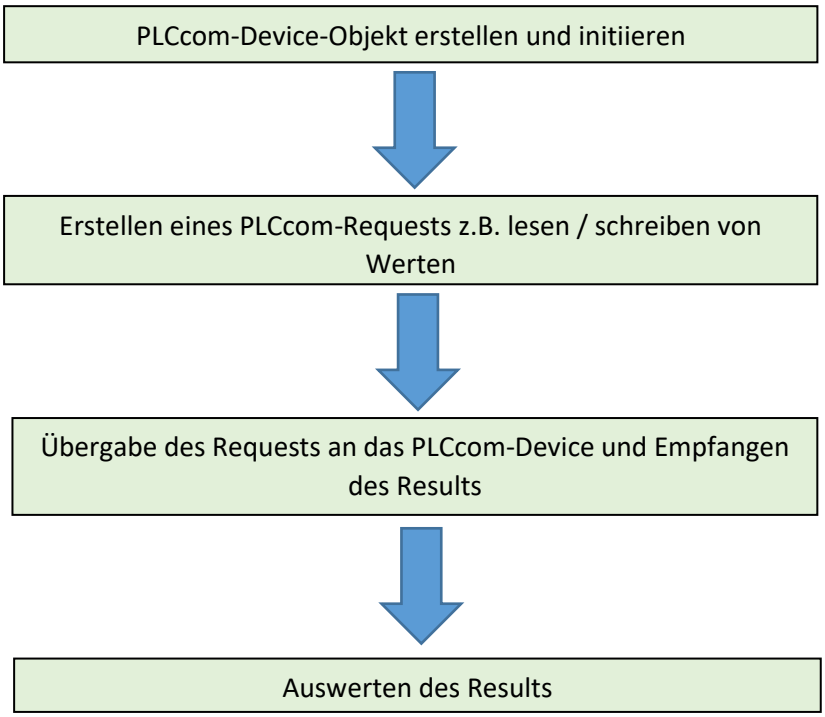
Java

```
authentication.User("your user name");
authentication.Serial("your user serial key");
```

Indi.An Systems GmbH
Flughafenallee 3
28199 Bremen
Deutschland
support@indi-systems.de
Tel + 49 421-989703-30
Fax + 49 421-989703-39

Erste Programmier-Schritte mit PLCcom

Die Einbindung der PLCcom-Library ist denkbar einfach und mit wenigen Codezeilen erledigt. Nachfolgend sind die notwendigen Implementierungsschritte aufgeführt.



Syntax: PLCcom-Device-Objekt erstellen und initiieren

CSharp

```
PLCcomDevice Device = new TCP_ISO_Device("192.168.1.100", 0, 2,
                                         ePLCType.S7_300_400_compatibel);
```

Visual Basic

```
Dim Device As PLCcomDevice = New TCP_ISO_Device("192.168.1.2", 0, 2, _
                                                ePLCType.S7_300_400_compatibel)
```

Java

```
PLCcomDevice Device = new TCP_ISO_Device("192.168.1.100", 0, 2,
ePLCType.S7_300_400_compatibel);
```

Syntax: Erstellen eines PLCcom-Requests z.B. Lesen von Werten

CSharp

```
//read 10 Bytes from DB1 at address 100
ReadDataRequest myReadDataRequest = new ReadDataRequest(
                    eRegion.DataBlock,  //Region
                    1,                  //DB only for datablock operations otherwise 0
                    0,                  //read start adress
                    eDataType.BYTE,     //desired datatype
                    10);                //Quantity of reading values
```

Visual Basic

```
'read 10 Bytes from DB1 at address 100
Dim myReadDataRequest As ReadDataRequest = new ReadDataRequest(
                    eRegion.DataBlock,
                    1,
                    0,
                    eDataType.BYTE,
                    10):
```

Java

```
//read 10 Bytes from DB1 at address 100
ReadDataRequest myReadDataRequest = new ReadDataRequest(eRegion.DataBlock,  //Region
                                    1, //DB only for datablock operations otherwise 0
                                    100, //read start adress
                                    eDataType.BYTE,  //desired datatype
                                    10); //Quantity of reading values
```

Indi.An Systems GmbH
Flughafenallee 3
28199 Bremen
Deutschland
support@indi-systems.de
Tel + 49 421-989703-30
Fax + 49 421-989703-39

Syntax: Übergabe des Requests an das PLCcom-Device und Empfangen des Results

CSharp

```
ReadDataResult res = Device.ReadData(myReadDataRequest);
```

Visual Basic

```
Dim res As ReadDataResult = Device.ReadData(myReadDataRequest)
```

Java

```
ReadDataResult res = Device.readData(myReadDataRequest);
```

Syntax: Auswerten des Requests

```
if (res.Quality == OperationResult.eQuality.GOOD)
{
   int Position = 0;
   foreach (Object item in res.GetValues())
    {
        Console.WriteLine("read Byte " + Position++.ToString() + " " + item.ToString());
    }
}
```

CSharp

Visual Basic

```
If res.Quality = OperationResult.eQuality.GOOD Then
   Dim sb As New StringBuilder()
   For Each item As ReadValue In res.FetchValues()
       sb.Append("Address " & item.Address.ToString() & " / Pos" &
       item.AddressPosition.ToString())
       sb.Append(" >> ")
       sb.Append(item.ToString())
       sb.Append(Environment.NewLine)
   Next
   Console.WriteLine(sb.ToString())
 End If
```

Indi.An Systems GmbH
Flughafenallee 3
28199 Bremen
Deutschland
support@indi-systems.de
Tel + 49 421-989703-30
Fax + 49 421-989703-39

Java

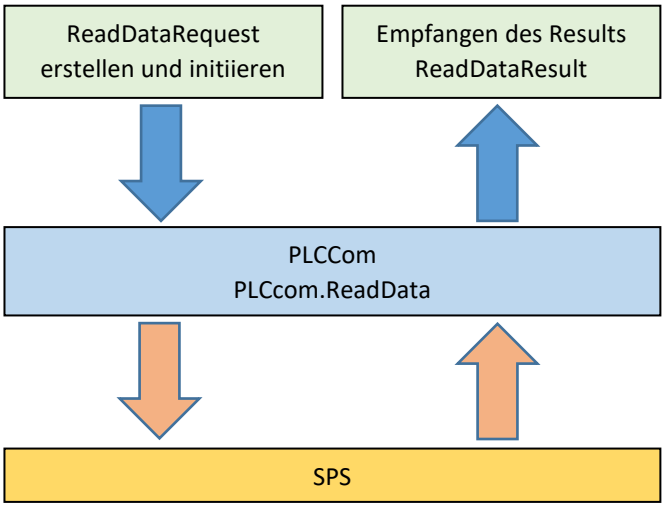
```
//evaluate results
if (res.Quality() == OperationResult.eQuality.GOOD) {
   int Position = 0;
   for (Object item : res.getValues()) {
     System.out.println(item.toString());
   }
}
```

Indi.An Systems GmbH
Flughafenallee 3
28199 Bremen
Deutschland
support@indi-systems.de
Tel + 49 421-989703-30
Fax + 49 421-989703-39

Funktionsübersicht

Einfaches Lesen von Daten aus der SPS

Das Lesen der Daten wird über einen ReadDataRequest initialisiert, mit dem Befehl ReadData ausgelöst. Das Ergebnis wird als ReadDataResult zurückgegeben.



Beispiel:

CSharp

```
PLCcomDevice Device = new TCP_ISO_Device("192.168.1.100", 0, 2,
                                         ePLCType.S7_300_400_compatibel);
//read 10 Bytes from DB1
ReadDataRequest myReadDataRequest = new ReadDataRequest(
                     eRegion.DataBlock,  //Region
                     1,                  //DB only for datablock operations otherwise 0
                     0,                  //read start adress
                     eDataType.BYTE,     //desired datatype
                     10);                //Quantity of reading values


ReadDataResult res = Device.ReadData(myReadDataRequest);


if (res.Quality == OperationResult.eQuality.GOOD)
{
   int Position = 0;
   foreach (Object item in res.GetValues())
    {
       Console.WriteLine("read Byte " + Position++.ToString() + " " + item.ToString());
    }
}
```

Indi.An Systems GmbH
Flughafenallee 3
28199 Bremen
Deutschland
support@indi-systems.de
Tel + 49 421-989703-30
Fax + 49 421-989703-39

Visual Basic

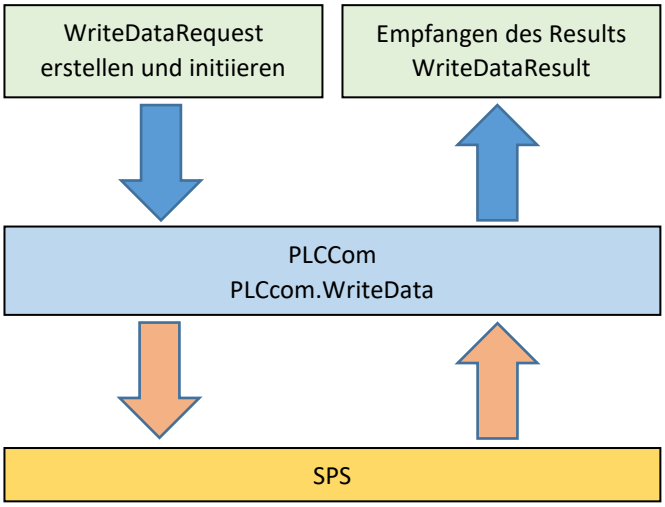
```
Dim Device As PLCcomDevice = New TCP_ISO_Device("192.168.1.2", 0, 2,
                                    ePLCType.S7_300_400_compatibel

read 10 Bytes from DB1
Dim myReadDataRequest As ReadDataRequest = new ReadDataRequest(
                      eRegion.DataBlock,
                      1,
                      0,
                      eDataType.BYTE,
                      10);

Dim res As ReadDataResult = Device.ReadData(myReadDataRequest)


If res.Quality = OperationResult.eQuality.GOOD Then
   For Each item As ReadValue In res.FetchValues()
       Console.WriteLine(item.ToString())
   Next
End If
```

Java

```
PLCcomDevice Device = new TCP_ISO_Device("192.168.1.100", 0, 2, ePLCType.S7_300_400_compatibel);

//read 10 Bytes from DB1
ReadDataRequest myReadDataRequest = new ReadDataRequest(eRegion.DataBlock,  //Region
                                    1, //DB only for datablock operations otherwise 0
                                    0, //read start adress
                                    eDataType.BYTE,  //desired datatype
                                    10); //Quantity of reading values

ReadDataResult res = Device.readData(myReadDataRequest);

//evaluate results
if (res.Quality() == OperationResult.eQuality.GOOD) {
   int Position = 0;
   for (Object item : res.getValues()) {
     System.out.println(item.toString());
   }
}
```

Einfaches Schreiben von Daten in die SPS

Das Schreiben von Daten wird über einen WriteDataRequest initialisiert, mit dem Befehl WriteData ausgelöst. Das Ergebnis wird als WriteDataResult zurückgegeben.



Beispiel:

CSharp

```
//declare a WriteRequest object and
//set the request parameters
WriteDataRequest myWriteRequest = new WriteDataRequest(eRegion.DataBlock, //Region
                                                       100,                //DB
                                                       0);                 //startaddress
//add writable Data here
//in  this case => write 4 bytes in DB100
myWriteRequest.addByte(new byte[] { 11, 12, 13, 14 });


//write
WriteDataResult res = Device.WriteData(myWriteRequest);

//evaluate results
if (res.Quality.Equals(OperationResult.eQuality.GOOD))
   {
       Console.WriteLine("Write successfull! Message: " + res.Message);
   }
```

Indi.An Systems GmbH
Flughafenallee 3
28199 Bremen
Deutschland
support@indi-systems.de
Tel + 49 421-989703-30
Fax + 49 421-989703-39

Visual Basic

```
'declare a WriteRequest object and
'set the request parameters
'Region
'DB / only for datablock operations otherwise 0
'write start adress
Dim myWriteRequest As New WriteDataRequest(eRegion.DataBlock, _
                                           100, _
                                           0)

'add writable Data here
'in  this case => write 4 bytes in DB100
myWriteRequest.addByte(New Byte() {11, 12, 13, 14})


'write
Dim res As WriteDataResult = Device.WriteData(myWriteRequest)

'evaluate results
If res.Quality.Equals(OperationResult.eQuality.GOOD) Then
    Console.WriteLine("Write successfull! Message: " + res.Message)
End If
```

Java

```
//declare a WriteRequest object and
//set the request parameters
WriteDataRequest myWriteRequest = new WriteDataRequest(eRegion.DataBlock,  //Region
                                                       100,                 //DB
                                                       0);                 //write start address

//add writable Data here
//in  this case => write 4 bytes in DB100
myWriteRequest.addByte(new byte[]{11, 12, 13, 14});

//write
System.out.println("begin write...");
WriteDataResult res = Device.writeData(myWriteRequest);

//evaluate results
if (res.Quality().equals(OperationResult.eQuality.GOOD)) {
    System.out.println("Write successfull! Message: " + res.Message());
}
```

Optimiertes Lesen und Schreiben von Daten

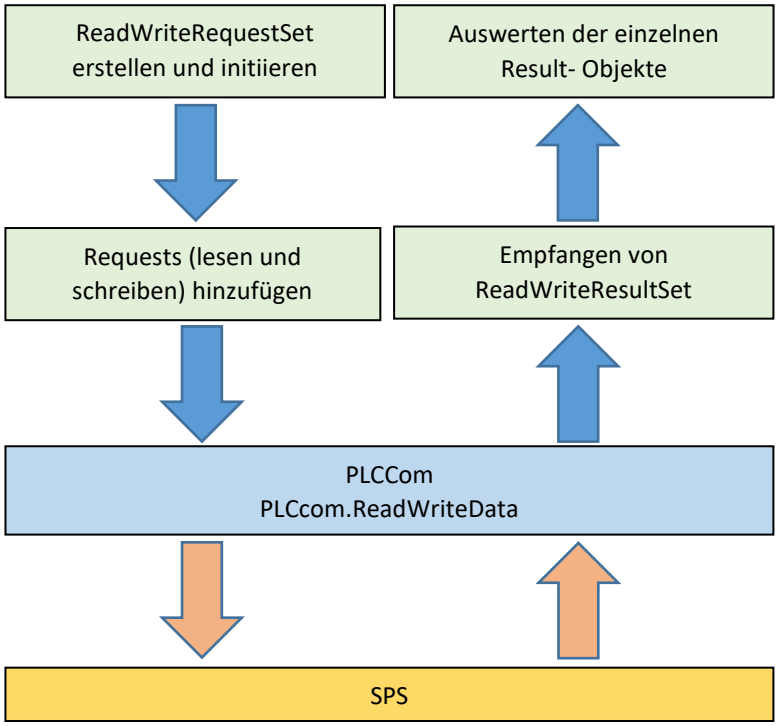
Zum optimierten Lesen und Schreiben werden die Lese- und Schreibzugriffe in einem ReadWriteRequestSet -Objekt zusammen gefasst.

Der optimierte Zugriff auf die SPS.Daten wird über einen oder mehrere ReadDataRequest- und / oder WriteDataRequest -Objekte initialisiert.

Diese Requests werden an das ReadWriteRequestSet-Object übergeben und mit dem Befehl ReadWriteData ausgelöst und auf Wunsch optimiert verarbeitet.

Das Ergebnis der Operation wird dem Entwickler als ReadWriteResultSet zurückgegeben, hier befinden sich die einzelnen ReadDataResult- und / oder WriteDataResult -Objekte zur weiteren Auswertung.

Funktion nicht für PPI-Verbindungen verfügbar, bitte beachten Sie die Funktionsübersicht auf unserer Website www.plccom.de



Optimierungsmöglichkeiten

Das ReadWriteRequestSet -Object wurde mit diversen Parametrierungs- und Optimierungsmöglichkeiten ausgestattet.

Über die Methode **RequestSet.SetOperationOrder(eOperationOrder)** kann festgelegt werden ob zuerst die Write- oder Read- Zugriffe durchgeführt werden. Per Default werden die Daten zuerst geschrieben und dann gelesen.

Zum Parametrieren der durchzuführenden Datenoptimierung dienen die Methoden:

RequestSet.SetReadOptimizationMode(eReadOptimizationMode) und **RequestSet.SetWriteOptimizationMode(eWriteOptimizationMode).** Per Default sind die Optimierungen abgeschaltet „NONE“.

Das enum eReadOptimizationMode besitzt die folgenden Member:

- NONE:
  Keine Optimierung, alle Read Requests werden nacheinander gelesen. Sicher aber langsam.
- CROSS_AREAS:
  Im CROSS_AREAS Modus werden die Lese- Anforderungen bereichsübergreifend zusammengeführt. Vorteil: Fragmentierte Bereiche (z. B. Daten über mehrere Datenblöcke) können gleichzeitig gelesen und geschrieben werden
- COMBINE_AREAS:
  Im COMBINE_AREAS Modus werden Lese-Anforderungen aus denselben Bereichen kombiniert. Vorteil: Schneller und performanter Zugriff auf Daten derselben Bereiche (z. B. Daten in demselben Datenblock)
- AUTO:
  PLCcom wählt für Sie automatisch die beste Optimierungsmethode für die Verarbeitung der Read Requests aus. Es werden nur die minimal erforderlichen SPS-Zugriffe durchgeführt.
  AUTO nur in der Expert Edition verfügbar

Das enum eWriteOptimizationMode besitzt die folgenden Member:

- NONE:
  Keine Optimierung, alle Write Requests werden nacheinander geschrieben. Sicher aber langsam.
- CROSS_AREAS:
  Im CROSS_AREAS Modus werden die Schreib- Anforderungen bereichsübergreifend zusammengeführt. Vorteil: Fragmentierte Bereiche (z. B. Daten über mehrere Datenblöcke) können gleichzeitig gelesen und geschrieben werden

Beispiel:

CSharp

```
PLCcomDevice Device = new TCP_ISO_Device("192.168.1.100", 0, 2,
                                         ePLCType.S7_300_400_compatibel);

ReadWriteRequestSet myRequestSet = new ReadWriteRequestSet();

//set optimize options
myRequestSet.SetOperationOrder(eOperationOrder.WRITE_BEVOR_READ);
myRequestSet.SetReadOptimizationMode(eReadOptimizationMode.AUTO);
myRequestSet.SetWriteOptimizationMode(eWriteOptimizationMode.CROSS_AREAS);

//declare a ReadRequest object set the request parameters,
//in this case => read 10 Bytes from DB1 at Byte 0
ReadDataRequest myReadDataRequest = new ReadDataRequest(
                    eRegion.DataBlock,  //Region
                    1,                  //DB only for datablock operations otherwise 0
                    0,                  //read start adress
                    eDataType.BYTE,     //desired datatype
                    10);                //Quantity of reading values

//add the read request to the request set
myRequestSet.AddRequest(myReadDataRequest);

//declare a WriteRequest object set the request parameters,
//in this case => write 4 bytes to DB100 at address 0
WriteDataRequest myWriteRequest = new WriteDataRequest(eRegion.DataBlock, //Region
                                                       100,               //DB
                                                       0);                //startaddress
//add writable Data here
//in  this case => write 4 bytes in DB100
myWriteRequest.addByte(new byte[] { 11, 12, 13, 14 });


//add the write request to the request set
myRequestSet.AddRequest(myWriteRequest);

//....... add more requests to request set

//read, write and getting the results
ReadWriteResultSet results = Device.ReadWriteData(myRequestSet);

// evaluate the results of read operations...
foreach (ReadDataResult res in results.GetReadDataResults())
{
   //for getting read results see chapter simple read
}

//...and evaluate the results of write operations
foreach (WriteDataResult res in results.GetWriteDataResults())
{
   //for getting write results see chapter simple write
}
```

Visual Basic

```
Dim Device As PLCcomDevice = New TCP_ISO_Device("192.168.1.100", 0, 2,
ePLCType.S7_300_400_compatibel)

Dim myRequestSet As ReadWriteRequestSet = New ReadWriteRequestSet()

//set optimize options
myRequestSet.SetOperationOrder(eOperationOrder.WRITE_BEVOR_READ)
myRequestSet.SetReadOptimizationMode(eReadOptimizationMode.AUTO)
myRequestSet.SetWriteOptimizationMode(eWriteOptimizationMode.CROSS_AREAS)


'declare a ReadRequest object set the request parameters,
'in this case => read 10 Bytes from DB1 at Byte 0
Dim myReadDataRequest As ReadDataRequest = New ReadDataRequest(eRegion.DataBlock, _
                                                               1, _
                                                               0, _
                                                               eDataType.[BYTE], _
                                                               10)
'add the read request to the request set
myRequestSet.AddRequest(myReadDataRequest)

Dim myWriteRequest As WriteDataRequest = New WriteDataRequest(eRegion.DataBlock, 100, 0)

'add writable Data here
'in  this case => write 4 bytes in DB100
myWriteRequest.addByte(New Byte() { 11, 12, 13, 14 })

'add the write request to the request set
myRequestSet.AddRequest(myWriteRequest)

'....... add more requests to request set

'read, write and getting the results
Dim results As ReadWriteResultSet = Device.ReadWriteData(myRequestSet)

'evaluate results

'evaluate the results of read operations...
For Each res As ReadDataResult In results.GetReadDataResults()


Next

'...and evaluate the results of write operations
For Each res As WriteDataResult In results.GetWriteDataResults()

Next
```

Indi.An Systems GmbH
Flughafenallee 3
28199 Bremen
Deutschland
support@indi-systems.de
Tel + 49 421-989703-30
Fax + 49 421-989703-39

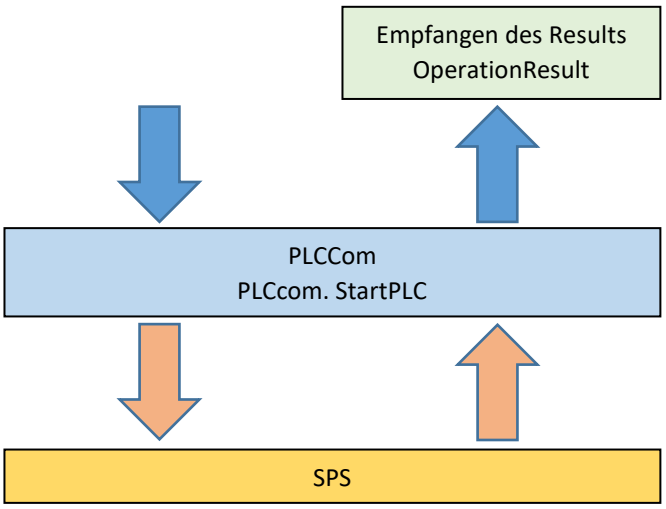
Java

```
PLCcomDevice Device = new TCP_ISO_Device("192.168.1.100", 0, 2, ePLCType.S7_300_400_compatibel);

ReadWriteRequestSet myRequestSet = new ReadWriteRequestSet();

//set optimize options
myRequestSet.setOperationOrder(eOperationOrder.WRITE_BEVOR_READ);
myRequestSet.setReadOptimizationMode(eReadOptimizationMode.AUTO);
myRequestSet.setWriteOptimizationMode(eWriteOptimizationMode.CROSS_AREAS);


// declare a ReadRequest object set the request parameters, //in this case =>
// read 10 Bytes from DB1 at Byte 0
ReadDataRequest myReadDataRequest = new ReadDataRequest(eRegion.DataBlock, // Region
              1, // DB only for datablock operations otherwise 0
              0, // read start adress
              eDataType.BYTE, // desired datatype
              10); // Quantity of reading values

// add the read request to the request set
myRequestSet.addRequest(myReadDataRequest);

// declare a WriteRequest object set the request parameters, //in this case =>
// write 4 bytes to DB100 at address 0
WriteDataRequest myWriteRequest = new WriteDataRequest(eRegion.DataBlock, // Region
              100, // DB
              0); // startaddress

// add writable Data here
// in this case => write 4 bytes in DB100
myWriteRequest.addByte(new byte[] { 11, 12, 13, 14 });

// add the write request to the request set
myRequestSet.addRequest(myWriteRequest);

// ..... add more requests to request set

// read, write and getting the results
ReadWriteResultSet results = Device.readWriteData(myRequestSet);

// evaluate the results of read operations...
for (ReadDataResult res : results.getReadDataResults()) {
       // for getting read results see chapter simple read
}
// ...and evaluate the results of write operations
for (WriteDataResult res : results.getWriteDataResults()) {
       // for getting write results see chapter simple write
}
```

SPS starten

Die SPS kann über den Befehl StartPLC gestartet werden. Es wird ein OperationResult-Objekt zurückgegeben.

Funktion nicht für alle CPUs verfügbar, bitte beachten Sie die Funktionsübersicht auf unserer Website www.plccom.de



Beispiel:

CSharp

```
OperationResult res = Device.StartPLC();
```

Visual Basic

```
Dim res As OperationResult = Device.StartPLC()
```

Java

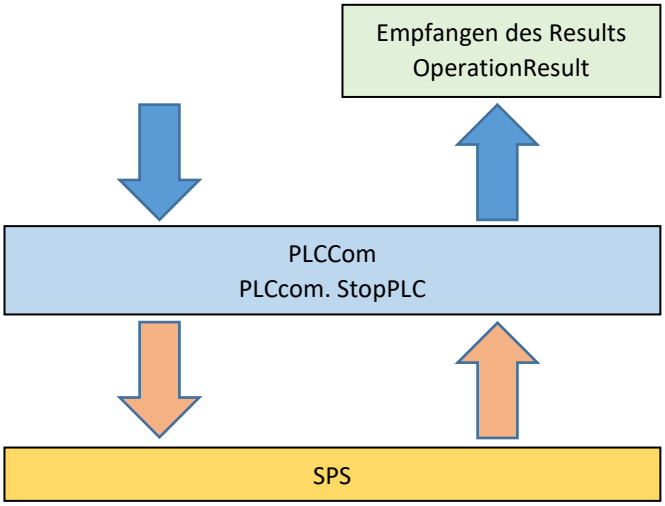
```
OperationResult res = Device.startPLC();
```

Indi.An Systems GmbH
Flughafenallee 3
28199 Bremen
Deutschland
support@indi-systems.de
Tel + 49 421-989703-30
Fax + 49 421-989703-39

SPS stoppen

Die SPS kann über den Befehl StopPLC gestoppt werden. Es wird ein OperationResult-Objekt zurückgegeben.

Funktion nicht für alle CPUs verfügbar, bitte beachten Sie die Funktionsübersicht auf unserer Website www.plccom.de



Beispiel:

CSharp

```
OperationResult res = Device.StopPLC();
```

Visual Basic

```
Dim res As OperationResult = Device.StopPLC()
```

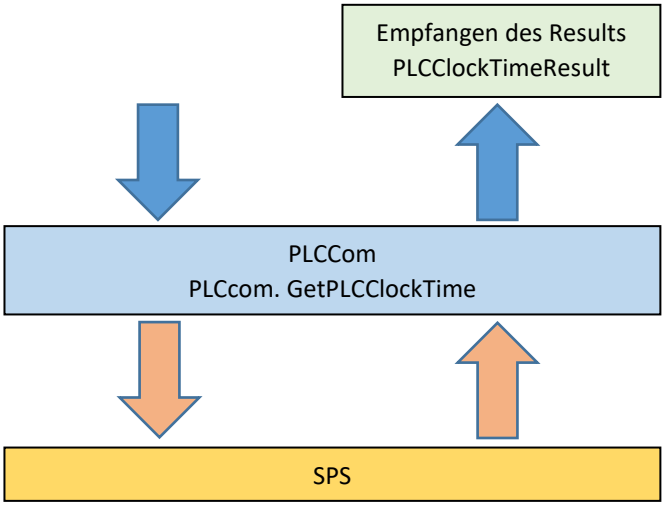
Java

```
OperationResult res = Device.stopPLC();
```

SPS Zeit lesen

Über den Befehl GetPLCClockTime kann die Zeit aus der SPS gelesen werden. Es wird ein PLCClockTimeResult-Objekt zurückgegeben.

Funktion nicht für alle CPUs verfügbar, bitte beachten Sie die Funktionsübersicht auf unserer Website www.plccom.de



Beispiel:

CSharp

```
PLCClockTimeResult res = Device.GetPLCClockTime();
```

Visual Basic

```
Dim res As PLCClockTimeResult = Device.GetPLCClockTime()
```

Java

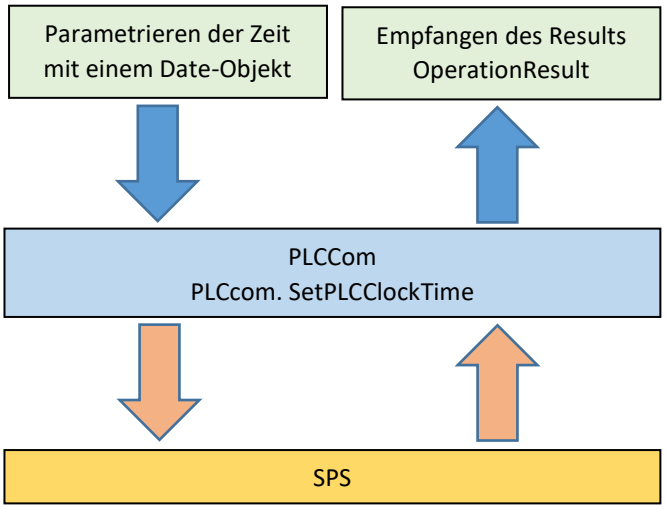
```
PLCClockTimeResult res = Device.getPLCClockTime();
```

Indi.An Systems GmbH
Flughafenallee 3
28199 Bremen
Deutschland
support@indi-systems.de
Tel + 49 421-989703-30
Fax + 49 421-989703-39

SPS Zeit setzen

Der Befehl SetPLCClockTime kann zum Setzen der Zeit innerhalb der SPS genutzt werden. Es wird ein Date-Objekt mit der gewünschten Zeit als Parameter übergeben. Es wird ein PLCClockTimeResult-Objekt zurückgegeben.

Funktion nicht für alle CPUs verfügbar, bitte beachten Sie die Funktionsübersicht auf unserer Website www.plccom.de



Beispiel:

CSharp

```
OperationResult res = Device.SetPLCClockTime(DateTime.Now);
```

Visual Basic

```
Dim res As OperationResult = Device.SetPLCClockTime(DateTime.Now)
```

Java

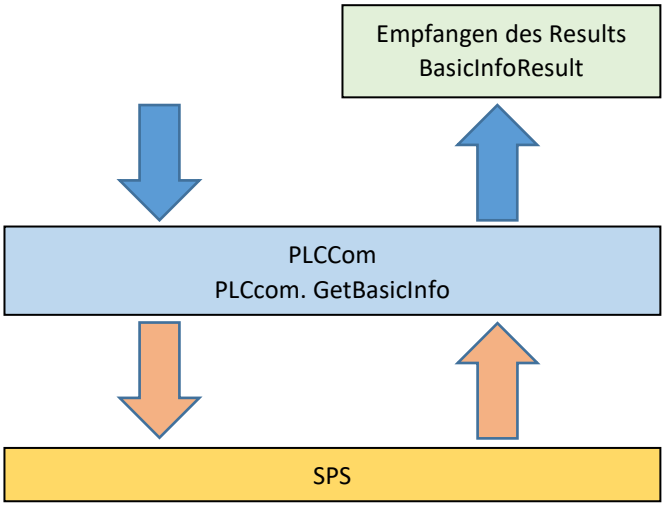
```
OperationResult res = Device.setPLCClockTime(Calendar.getInstance().getTime());
```

Indi.An Systems GmbH
Flughafenallee 3
28199 Bremen
Deutschland
support@indi-systems.de
Tel + 49 421-989703-30
Fax + 49 421-989703-39

SPS Basic-Infos auslesen

Mit dem Befehl GetBasicInfo lassen sich Kerninformationen wie z.B. Typ, Bestellnummer, Modul-Version und Firmware-Version der benutzen CPU auslesen. Es wird ein BasicInfoResult-Objekt zurückgegeben.

Funktion nicht für alle CPUs verfügbar, bitte beachten Sie die Funktionsübersicht auf unserer Website www.plccom.de



Beispiel:

CSharp

```
BasicInfoResult res = Device.GetBasicInfo();
```

Visual Basic

```
Dim res As BasicInfoResult = Device.GetBasicInfo()
```

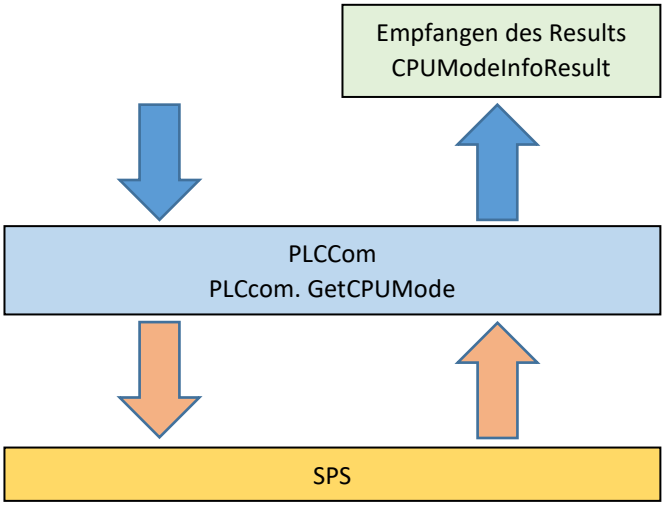
Java

```
BasicInfoResult res = Device.getBasicInfo();
```

CPU-Modus auslesen

Mit dem Befehl GetCPUMode lässt sich feststellen in welchem Modus z.B. Run, Startup, Stop, etc. sich die CPU befindet. Es wird ein CPUModeInfoResult-Objekt zurückgegeben.

Funktion nicht für alle CPUs verfügbar, bitte beachten Sie die Funktionsübersicht auf unserer Website www.plccom.de



Beispiel:

CSharp

```
CPUModeInfoResult res = Device.GetCPUMode();
```

Visual Basic

```
Dim res As CPUModeInfoResult = Device.GetCPUMode()
```

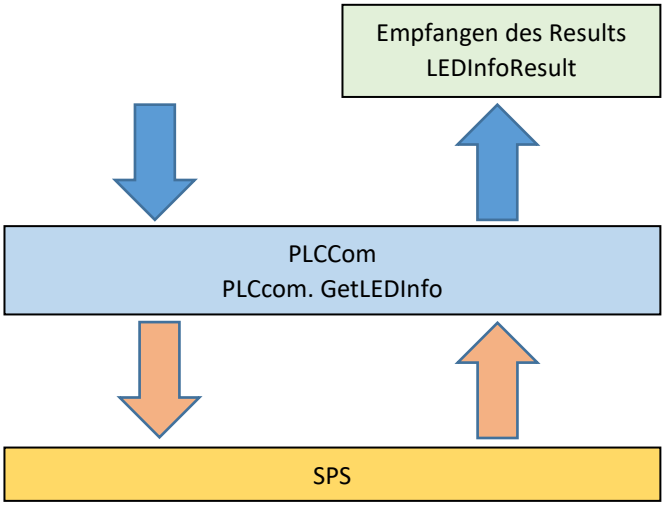
Java

```
CPUModeInfoResult res = Device.getCPUMode();
```

LED-Status auslesen

Der Befehl GetLEDInfo dient zur Abfrage des aktuellen Zustands der LEDs z.B. Ein, Aus, Blinken, etc. an der CPU. Es wird ein LEDInfoResult-Objekt zurückgegeben.

Funktion nicht für alle CPUs verfügbar, bitte beachten Sie die Funktionsübersicht auf unserer Website www.plccom.de



Beispiel:

CSharp

```
LEDInfoResult res = Device.GetLEDInfo();
```

Visual Basic

```
Dim res As LEDInfoResult = Device.GetLEDInfo()
```

Java

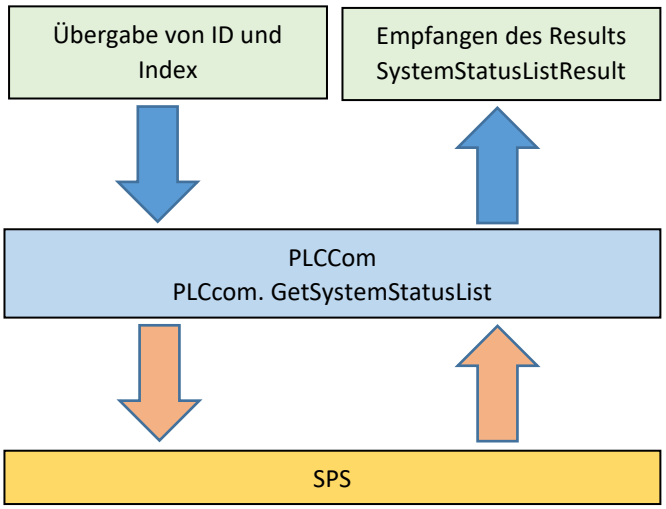
```
LEDInfoResult res = Device.getLEDInfo();
```

Indi.An Systems GmbH
Flughafenallee 3
28199 Bremen
Deutschland
support@indi-systems.de
Tel + 49 421-989703-30
Fax + 49 421-989703-39

Systemzustandslisten SZL lesen

Die Funktion GetSystemStatusList versetzt den Anwender in die Lage auf die Systemzustandsliste innerhalb der SPS zuzugreifen. Für weitere Informationen zur Systemzustandsliste kontaktieren Sie bitte den SPS-Hersteller oder schlagen innerhalb der Dokumentation nach. Es wird ein SystemStatusListResult -Objekt zurückgegeben.

Funktion nicht für alle CPUs verfügbar, bitte beachten Sie die Funktionsübersicht auf unserer Website www.plccom.de



Beispiel:

CSharp

```
SystemStatusListResult res = Device.GetSystemStatusList(SSL_ID, SSL_Index);
```

Visual Basic

```
Dim res As SystemStatusListResult = Device.GetSystemStatusList(SSL_ID, SSL_Index)
```

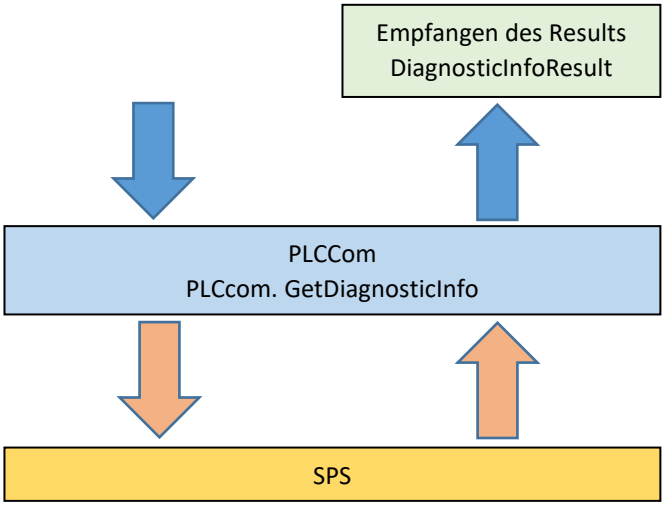
Java

```
SystemStatusListResult res = Device.getSystemStatusList(SSL_ID, SSL_Index);
```

Diagnosedaten lesen

Der Befehl GetDiagnosticInfo dient zur Abfrage der aktuellen Diagnosedatenliste aus der CPU. Es wird ein DiagnosticInfoResult-Objekt zurückgegeben.

Funktion nicht für alle CPUs verfügbar, bitte beachten Sie die Funktionsübersicht auf unserer Website www.plccom.de



Beispiel:

CSharp

```
DiagnosticInfoResult res = Device.GetDiagnosticInfo();
```

Visual Basic

```
Dim res As DiagnosticInfoResult = Device.GetDiagnosticInfo()
```

Java

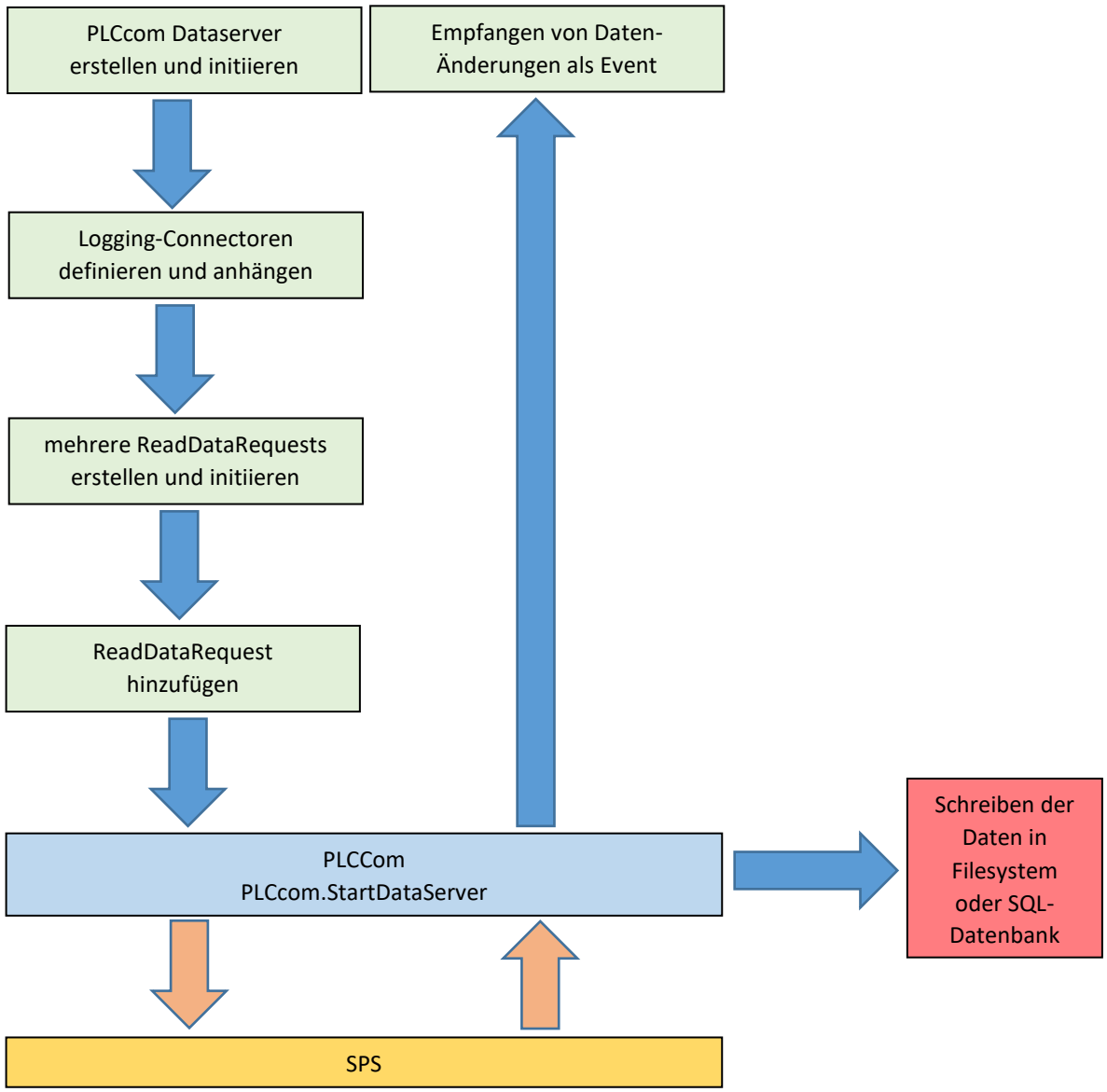
```
DiagnosticInfoResult res = Device.getDiagnosticInfo();
```

Indi.An Systems GmbH
Flughafenallee 3
28199 Bremen
Deutschland
support@indi-systems.de
Tel + 49 421-989703-30
Fax + 49 421-989703-39

PLCCom-DataServer (nur in der Expert-Version)

Mittels dem PLCcom-Dataserver wird der Anwender in die Lage versetzt, definierte Speicherbereiche in der SPS zu überwachen. Bei Veränderung der Daten wird ein Event ausgelöst, wodurch der Benutzer über die Veränderung unterrichtet wird. Die Kommunikation zur SPS wird optimiert durchgeführt, sodass die Anzahl der benötigten Zugriffe auf ein Minimum reduziert wird.

Zusätzlich ist es möglich Connectoren für das Filesystem oder für SQL-Datenbanken zu definieren, sodass die Daten fortlaufend geschrieben werden. Auch ist es möglich ein aktuelles Datenabbild im Filesystem oder in einer SQL-Datenbank zur Verfügung zu stellen. Zur erweiterten Sicherheit lassen sich die Daten im Filesystem auch verschlüsselt ablegen, zur Entschlüsselung stehen Methoden zur Verfügung.



Beispiel:

CSharp

```
PLCcom.PLCComDataServer.PLCComDataServer myDataServer = null;
PLCcomDevice Device = null;

Console.WriteLine("Start Connect to TCPIP device...");
//Create an PLCcom-Device instance
Device = new TCP_ISO_Device("192.168.1.100", 0, 2, ePLCType.S7_300_400_compatibel);

//set autoconnect to true and idle time till disconnect to 10000 milliseconds
Device.setAutoConnect(true, 10000);

//Create an instance depending on the device type
Console.WriteLine("Create DataServer PLCDataServerTCP1...");
myDataServer = new PLCcom.PLCComDataServer.PLCComDataServer_TCP("PLCDataServerTCP1", (TCP_ISO_Device)Device, 500);

//register incoming events
//register events
myDataServer.OnConnectionStateChange += new
PLCcom.PLCComDataServer.PLCComDataServer.ConnectionStateChangeEventHandler(myDataServer_OnConnectionStateChange);
myDataServer.OnReadDataResultChange += new
PLCcom.PLCComDataServer.PLCComDataServer.ReadDataResultChangeEventHandler(myDataServer_OnReadDataResultChange);
myDataServer.OnIncomingLogEntry += new
PLCcom.PLCComDataServer.PLCComDataServer.OnIncomingLogEntryDelegate(myDataServer_OnIncomingLogEntry);

//define new request
Console.WriteLine("Create new Request Read 4 Bytes from DB1 at address 0 ...");
ReadDataRequest RequestItem1 = new ReadDataRequest(eRegion.DataBlock,          //Region
                                                   1,                          //datablock
                                                   0,                          //startAdress
                                                   eDataType.BYTE,             //target data type
                                                   4);                         //Quantity

//add new request to plccom data server
myDataServer.AddReadDataRequest(RequestItem1, "Request1");

//define new request
Console.WriteLine("Create new Request Read 10 DWord from Flags_Markers at address 4 ...");
ReadDataRequest RequestItem2 = new ReadDataRequest(eRegion.Flags_Markers,      //Region
                                                   0,                          //datablock
                                                   4,                          //startAdress
                                                   eDataType.DWORD,            //target data type
                                                   4);                         //Quantity

//add new request to plccom data server
myDataServer.AddReadDataRequest(RequestItem2, "Request2");


//add on or more Loggingconnector with logging and writing of a data image into filesystem or database
//in this case create a new FileSystemConnector instanc
PLCcom.ExternalLogging.LoggingConnector con = new
         PLCcom.ExternalLogging.FileSystemConnector(System.Threading.Thread.GetDomain().BaseDirectory, //Target folder
         "FileSystemConnector1",                                 //unique connector name
         ';',                                                    //text separator recommendation ';'
         true,                                                   //activate progressive logging
         true,                                                   //activate immage writing
         PLCcom.ExternalLogging.eImageOutputFormat.dat,          //output format .dat or .xml
         10,                                                     //restrict the maximum number of files. -1 = Disabled.
         24,                                                     //restrict the maximum age of files. -1 = Disabled.
         30,                                                     //You can restrict the maximum size of files. -1 = Disabled.
         string.Empty); //If you enter an encryption password, the data is stored in encrypted form.
//add Connector to Dataserver
myDataServer.AddOrReplaceLoggingConnector(con);

//start PLCcom data server
Console.ReadLine();
//stop PLCcom data server
myDataServer.StopServer();
```

Visual Basic

```
Dim myDataServer As PLCcom.PLCComDataServer.PLCComDataServer = Nothing
Dim Device As PLCcomDevice = Nothing

Console.WriteLine("Start Connect to TCPIP device...")
'Create an PLCcom-Device instance
Device = New TCP_ISO_Device("192.168.1.2", 0, 2, ePLCType.S7_300_400_compatibel)

'set autoconnect to true and idle time till disconnect to 10000 milliseconds
Device.setAutoConnect(True, 10000)

'Create an instance depending on the device type
Console.WriteLine("Create DataServer PLCDataServerTCP1...")
myDataServer = New PLCcom.PLCComDataServer.PLCComDataServer_TCP("DataServerTCP1", DirectCast(Device, TCP_ISO_Device), 500)

'register incoming events
AddHandler myDataServer.OnConnectionStateChange, AddressOf myDataServer_OnConnectionStateChange
AddHandler myDataServer.OnReadDataResultChange, AddressOf myDataServer_OnReadDataResultChange
AddHandler myDataServer.OnIncomingLogEntry, AddressOf myDataServer_OnIncomingLogEntry

Console.WriteLine("Create new Request Read 4 Bytes from DB1 at address 0 ...")
'Parameter:
'-Region
'-datablock
'-startAdress
'-target data type
'-Quantity
Dim RequestItem1 As New ReadDataRequest(eRegion.DataBlock, 1, 0, eDataType.[BYTE], 4)
'add new request to plccom data server
myDataServer.AddReadDataRequest(RequestItem1, "Request1")

Console.WriteLine("Create new Request Read 10 DWord from Flags_Markers at address 4 ...")
'Parameter:
'-Region
'-datablock
'-startAdress
'-target data type
'-Quantity
Dim RequestItem2 As New ReadDataRequest(eRegion.Flags_Markers, 0, 4, eDataType.DWORD, 4)
'add new request to plccom data server
myDataServer.AddReadDataRequest(RequestItem2, "Request2")

'add on or more Loggingconnector with logging and writing of a data image into filesystem or database. Parameter:
'-Target folder
'-unique connector name
'-text separator recommendation ';'
'-activate progressive logging
'-activate immage writing
'-output format .dat or .xml
'-restrict the maximum number of files. When the value is exceeded the old files are automatically deleted. -1 = Disabled.
'-restrict the maximum age of files. When the value is exceeded the old files are automatically deleted. -1 = Disabled.
'-You can restrict the maximum size of files. When the value is exceeded the old files are deleted. -1 = Disabled.
'-If you enter an encryption password, the data is stored in encrypted form.
Dim con As PLCcom.ExternalLogging.LoggingConnector = New
PLCcom.ExternalLogging.FileSystemConnector(System.Threading.Thread.GetDomain().BaseDirectory,
                                           "FileSystemConnector1", _
                                           ";"c, _
                                           True, _
                                           True, _
                                           PLCcom.ExternalLogging.eImageOutputFormat.dat, _
                                           10, _
                                           24, _
                                           30, _
                                           String.Empty)
'add Connector to Dataserver
myDataServer.AddOrReplaceLoggingConnector(con)
'start PLCcom data server
myDataServer.StartServer()
Console.ReadLine()
'stop PLCcom data server
myDataServer.StopServer()
```

Indi.An Systems GmbH
Flughafenallee 3
28199 Bremen
Deutschland
support@indi-systems.de
Tel + 49 421-989703-30
Fax + 49 421-989703-39

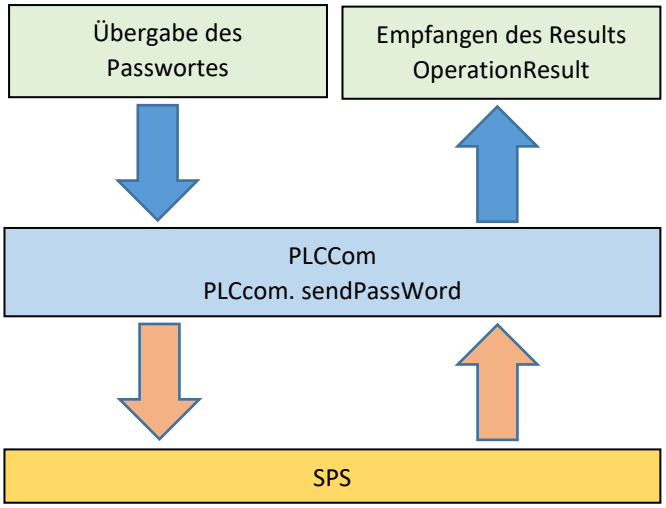
Java

```
PLCComDataServer myDataServer = null;
PLCcomDevice Device = null;

BufferedReader input = new BufferedReader(new InputStreamReader(System.in));
System.out.println("Start Connect to TCPIP device...");
//Create an PLCcom-Device instance
Device = new TCP_ISO_Device("192.168.1.100", 0, 2, ePLCType.S7_300_400_compatibel);

//set autoconnect to true and idle time till disconnect to 10000 milliseconds
Device.setAutoConnect(true, 10000);

//Create an instance depending on the device type
System.out.println("Create DataServer PLCDataServerTCP1...");
myDataServer = new PLCComDataServer_TCP("PLCDataServerTCP1", (TCP_ISO_Device)Device, 500);

//register incoming events
// register Connection state change event
myDataServer.connectionStateChangeNotifier = new ConnectionStateChangeNotifier(this);
// register incoming log event
myDataServer.incomingLogEntryEventNotifier = new IncomingLogEntryEventNotifier(this);
// register change ReadDataResult event
myDataServer.readDataResultChangeNotifier = new ReadDataResultChangeNotifier(this);

//define new request
System.out.println("Create new Request Read 4 Bytes from DB1 at address 0 ...");
ReadDataRequest RequestItem1 = new ReadDataRequest(eRegion.DataBlock,          //Region
                                                   1,                          //datablock
                                                   0,                          //startAdress
                                                   eDataType.BYTE,             //target data type
                                                   4);                         //Quantity

//add new request to plccom data server
myDataServer.addReadDataRequest(RequestItem1, "Request1");

//define new request
System.out.println("Create new Request Read 10 DWord from Flags_Markers at address 4 ...");
ReadDataRequest RequestItem2 = new ReadDataRequest(eRegion.Flags_Markers,      //Region
                                                   0,                          //datablock
                                                   4,                          //startAdress
                                                   eDataType.DWORD,            //target data type
                                                   4);                         //Quantity

//add new request to plccom data server
myDataServer.addReadDataRequest(RequestItem2, "Request2");

//add on or more Loggingconnector with logging and writing of a data image into filesystem or database
//in this case create a new FileSystemConnector instance
LoggingConnector con = new FileSystemConnector(new File(".").getAbsolutePath(), //Target folder
                                    "FileSystemConnector1",          //unique connector name
                                    ';',                             //text separator recommendation ';'
                                    true,                            //activate progressive logging
                                    true,                            //activate image writing
                                    eImageOutputFormat.dat,         //output format .dat or .xml
                                    10, //restrict the maximum number of files. -1 = Disabled.
                                    24, //restrict the maximum age of files. -1 = Disabled.
                                    30, //You can restrict the maximum size of files. -1 = Disabled.
                                    "");//If you enter an encryption password, the data is stored in encrypted form
//add Connector to Dataserver
myDataServer.addOrReplaceLoggingConnector(con);

//start PLCcom data server
myDataServer.startServer();
input.readLine();

//stop PLCcom data server
myDataServer.stopServer();
```

Passwort senden (nur in der Expert-Version)

Mit der Funktion sendPassWord lassen sich Passwortgeschütze SPSsen freischalten, vorausgesetzt der Anwender kennt das vergebene Passwort. Es wird ein OperationResult-Objekt zurückgegeben.

Funktion nicht für alle CPUs verfügbar, bitte beachten Sie die Funktionsübersicht auf unserer Website www.plccom.de



Beispiel:

CSharp

```
OperationResult res = Device.sendPassWord(“PW”);
```

Visual Basic

```
Dim res As OperationResult = Device.sendPassWord(“PW”)
```

Java

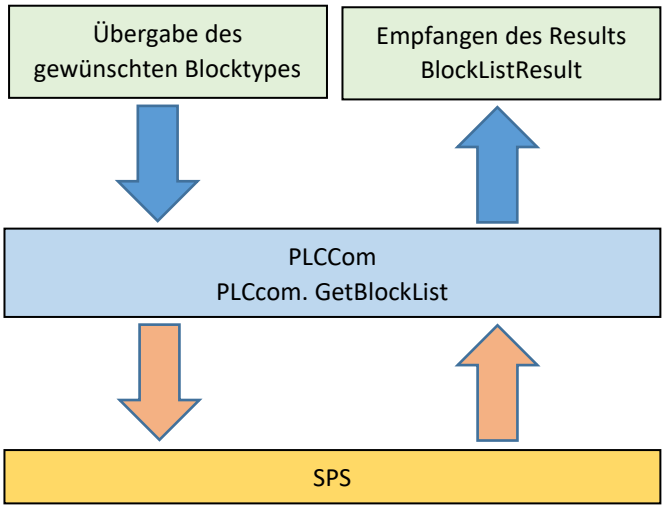
```
OperationResult res = Device.sendPassWord(“PW”);
```

Indi.An Systems GmbH
Flughafenallee 3
28199 Bremen
Deutschland
support@indi-systems.de
Tel + 49 421-989703-30
Fax + 49 421-989703-39

Objektliste aus SPS abrufen (nur in der Expert-Version)

Die Funktion GetBlockList versetzt den Anwender in die Lage mit einem Befehl die komplette Objektliste oder eine Liste eines bestimmten Blocktype wie z.B. Datablock von der SPS herunterzuladen. Es wird ein BlockListResult –Objekt mit weiteren Informationen zurückgegeben.

Funktion nicht für alle CPUs verfügbar, bitte beachten Sie die Funktionsübersicht auf unserer Website www.plccom.de



Beispiel:

CSharp

```
BlockListResult res = Device.GetBlockList(BlockType);
```

Visual Basic

```
Dim res As BlockListResult = Device.GetBlockList(BlockType)
```

Java

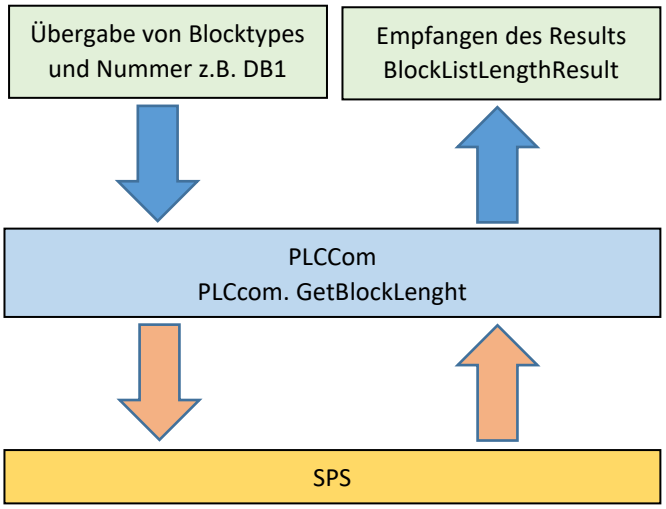
```
BlockListResult res = Device.getBlockList(BlockType);
```

Indi.An Systems GmbH
Flughafenallee 3
28199 Bremen
Deutschland
support@indi-systems.de
Tel + 49 421-989703-30
Fax + 49 421-989703-39

Länge eines Objektes abrufen (nur in der Expert-Version)

Mittels der Funktion GetBlockLenght kann der die Länge eines bestimmten SPS-Objektes erfragen. Die Längenangabe erfolgt in Bytes. Es wird ein BlockListLengthResult –Objekt mit weiteren Informationen zurückgegeben.

Funktion nicht für alle CPUs verfügbar, bitte beachten Sie die Funktionsübersicht auf unserer Website www.plccom.de



Beispiel:

CSharp

```
BlockListLengthResult res = Device.GetBlockLenght(BlockType, BlockNumber);
```

Visual Basic

```
Dim res As BlockListLengthResult = Device.GetBlockLenght(BlockType, BlockNumber)
```

Java

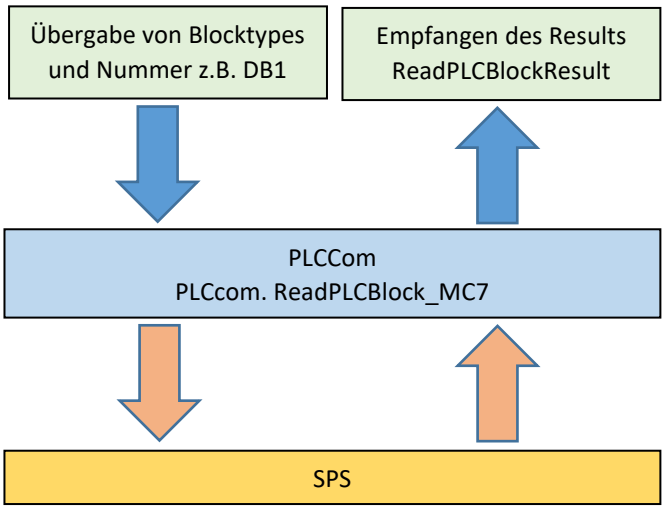
```
BlockListLengthResult res = Device.getBlockLenght(BlockType, BlockNumber);
```

Indi.An Systems GmbH
Flughafenallee 3
28199 Bremen
Deutschland
support@indi-systems.de
Tel + 49 421-989703-30
Fax + 49 421-989703-39

Backup eines Objektes (nur in der Expert-Version)

Mittels der Funktion ReadPLCBlock_MC7 kann der Code eines gewünschten SPS-Objektes im mc7-Format ausgelesen und abgespeichert werden. Des Weiteren werden Zusatzinformationen wie Code, Erstellungssprachen, Autor usw. ausgegeben. Es wird ein ReadPLCBlockResult–Objekt mit weiteren umfangreichen Informationen zurückgegeben.

Funktion nicht für alle CPUs verfügbar, bitte beachten Sie die Funktionsübersicht auf unserer Website www.plccom.de



Beispiel:

CSharp

```
eBlockType BlockType = eBlockType.FB;
int BlockNumber = 1;

//open SaveFileDialog
SaveFileDialog sfd = new SaveFileDialog();
sfd.Filter = "*.mc7|*.mc7|*.bin|*.bin|*.*|*'.*";
DialogResult dr = sfd.ShowDialog();
if (dr == DialogResult.OK)
{
   //read Block into ReadPLCBlockResult
   ReadPLCBlockResult res = Device.ReadPLCBlock_MC7(BlockType, BlockNumber);

  if (res.Quality == OperationResult.eQuality.GOOD)
     {
       //save buffer in specified file
       System.IO.FileStream fs = new System.IO.FileStream(sfd.FileName,
       System.IO.FileMode.Create, System.IO.FileAccess.Write);
       fs.Write(res.Buffer, 0, res.Buffer.Length);
       fs.Close();
       MessageBox.Show("Block " + BlockType.ToString() + BlockNumber.ToString() + "successful
       saved" + sfd.FileName, "", MessageBoxButtons.OK, MessageBoxIcon.Information);
       }
 }
```

Visual Basic

```
Dim BlockType As eBlockType = eBlockType.FB
Dim BlockNumber As Integer = 1

'open SaveFileDialog
Dim sfd As New SaveFileDialog()
sfd.Filter = "*.mc7|*.mc7|*.bin|*.bin|*.*|*'.*"
Dim dr As DialogResult = sfd.ShowDialog()
If dr = DialogResult.OK Then
'read Block into ReadPLCBlockResult
    Dim res As ReadPLCBlockResult = Device.ReadPLCBlock_MC7(BlockType, BlockNumber)

    If res.Quality = OperationResult.eQuality.GOOD Then
       'save buffer in specified file
       Dim fs As New System.IO.FileStream(sfd.FileName, System.IO.FileMode.Create, _
       System.IO.FileAccess.Write)
       fs.Write(res.Buffer, 0, res.Buffer.Length)
       fs.Close()
       MessageBox.Show(("Block " & BlockType.ToString() & BlockNumber.ToString() &
       "successful_saved") + sfd.FileName, "", MessageBoxButtons.OK,
       MessageBoxIcon.Information)
    End If
End If
```

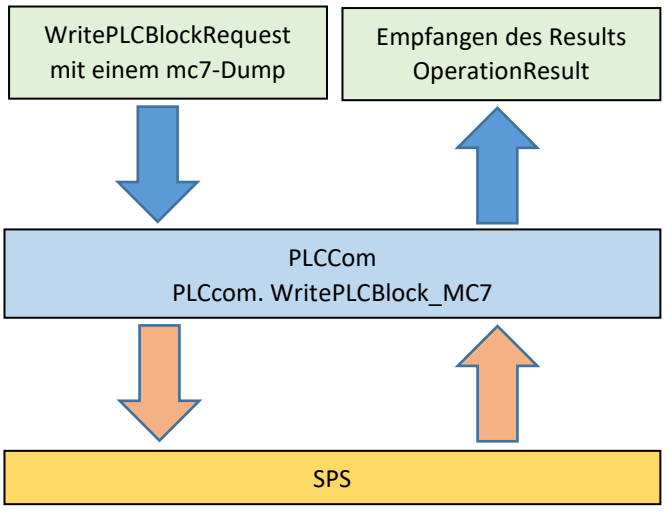
Java

```
eBlockType BlockType = eBlockType.FB;
int BlockNumber = 1;

// open SaveFileDialog
final JFileChooser dr = new JFileChooser(new File("."));
FileFilter filter = new FileNameExtensionFilter("Binary Files *.mc7", "mc7");
dr.addChoosableFileFilter(filter);
int returnVal = dr.showSaveDialog(this);
if (returnVal == JFileChooser.APPROVE_OPTION) {

	// read Block into ReadPLCBlockResult
	ReadPLCBlockResult res = Device.readPLCBlock_MC7(BlockType, BlockNumber);

	// evaluate values and write file
	if (res.Quality().equals(OperationResult.eQuality.GOOD)) {
		// save buffer in specified file
		File file = new File(dr.getSelectedFile().getAbsolutePath());

		// rename file to .mc7, you can adjust the extension
		if (!file.getAbsolutePath().endsWith(".mc7")) {
			file = new File(dr.getSelectedFile().getAbsolutePath() + ".mc7");
			// if file doesnt exists, then create it
			if (!file.exists()) {
				file.createNewFile();
			}
			FileOutputStream fs = new FileOutputStream(file);
			fs.write(res.getBuffer());
			fs.close();
		}
	}
}
```

Restore eines Objektes (nur in der Expert-Version)

Ergänzend zum Backup kann mit der Funktion WritePLCBlock_MC7 eine Backup-Datei wieder zurück in die SPS übertragen werden. Es ist auch möglich die Daten unter geänderter Blocknummer zurückzuspielen. Zur Übergabe der Daten wird ein WritePLCBlockRequest benutzt. Abschließend wird ein OperationResult–Objekt zurückgegeben.

Funktion nicht für alle CPUs verfügbar, bitte beachten Sie die Funktionsübersicht auf unserer Website www.plccom.de



Beispiel:

CSharp

```
OpenFileDialog ofd = new OpenFileDialog();
ofd.Filter = "*.mc7|*.mc7|*.bin|*.bin|*.*|*'.*";
DialogResult dr = ofd.ShowDialog();
if (dr == DialogResult.OK)
{
        System.IO.FileStream fs = new System.IO.FileStream(ofd.FileName,
        System.IO.FileMode.Open, System.IO.FileAccess.Read);

        byte[] buffer = new byte[fs.Length];
        fs.Read(buffer, 0, (int)fs.Length);
        fs.Close();

        WritePLCBlockRequest Requestdata = new WritePLCBlockRequest(buffer);
        //Write Buffer into PLC
        OperationResult res = Device.WritePLCBlock_MC7(Requestdata);

        if (res.Quality == OperationResult.eQuality.GOOD)
        {
                MessageBox.Show("Block " + Requestdata.BlockInfo.Header.BlockType.ToString() +
                Requestdata.BlockInfo.Header.BlockNumber.ToString() +
                resources.GetString("successful_saved_PLC") + ofd.FileName,
        }
}
```

Visual Basic

```
Dim ofd As New OpenFileDialog()
ofd.Filter = "*.mc7|*.mc7|*.bin|*.bin|*.*|*'.*"
Dim dr As DialogResult = ofd.ShowDialog()
If dr = DialogResult.OK Then
        Dim fs As New System.IO.FileStream(ofd.FileName, System.IO.FileMode.Open,
        System.IO.FileAccess.Read)
        Dim buffer As Byte() = New Byte(fs.Length - 1) {}
        fs.Read(buffer, 0, CInt(fs.Length))
        fs.Close()

        Dim Requestdata As New WritePLCBlockRequest(buffer)
        'Write Buffer into PLC
        Dim res As OperationResult = Device.WritePLCBlock_MC7(Requestdata)
        If res.Quality = OperationResult.eQuality.GOOD Then
                MessageBox.Show(("Block " & Requestdata.BlockInfo.Header.BlockType.ToString() &
                Requestdata.BlockInfo.Header.BlockNumber.ToString() &
                resources.GetString("successful_saved_PLC")) + ofd.FileName, "",
                MessageBoxButtons.OK, MessageBoxIcon.Information)
        End If
End If
```

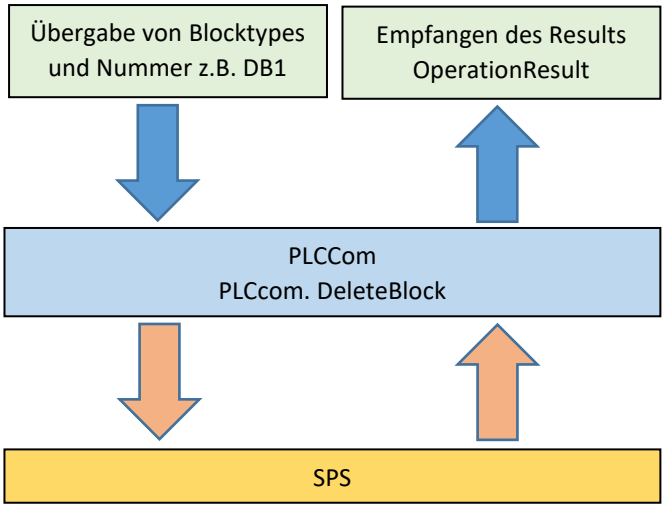
Java

```
// open SaveFileDialog
final JFileChooser dr = new JFileChooser(new File("."));
FileFilter filter = new FileNameExtensionFilter("Binary Files *.mc7", "mc7");
dr.addChoosableFileFilter(filter);
int returnVal = dr.showOpenDialog(this);
if (returnVal == JFileChooser.APPROVE_OPTION) {
InputStream is = null;
byte[] buffer = null;
try {
       is = new BufferedInputStream(new
       FileInputStream(dr.getSelectedFile().getAbsolutePath()));
       buffer = new byte[is.available()];
       is.read(buffer);
} finally {
       if (is != null) {
              is.close();
       }
}
WritePLCBlockRequest Requestdata = new WritePLCBlockRequest(buffer);
// Write Buffer into PLC
OperationResult res = Device.writePLCBlock_MC7(Requestdata);
```

Indi.An Systems GmbH
Flughafenallee 3
28199 Bremen
Deutschland
support@indi-systems.de
Tel + 49 421-989703-30
Fax + 49 421-989703-39

Löschen eines Objektes in der SPS (nur in der Expert-Version)

Mittels der Funktion DeleteBlock kann ein bestimmtes SPS-Objektes gelöscht werden.
Es wird ein OperationResult –Objekt mit weiteren Informationen zurückgegeben.
Funktion nicht für alle CPUs verfügbar, bitte beachten Sie die Funktionsübersicht auf unserer Website www.plccom.de



Beispiel:

CSharp

```
OperationResult res = Device.DeleteBlock(BlockType, BlockNumber);
```

Visual Basic

```
Dim res As OperationResult = Device.DeleteBlock(BlockType, BlockNumber)
```

Java

```
OperationResult res = Device.deleteBlock(bip.BlockType, bip.BlockNumber);
```

Indi.An Systems GmbH
Flughafenallee 3
28199 Bremen
Deutschland
support@indi-systems.de
Tel + 49 421-989703-30
Fax + 49 421-989703-39

Haben Sie Fragen?

Rufen Sie uns bitte unter der Hotline +49 421 98970330 an, oder senden Sie Ihre Frage an support@indi-systems.de.

Wir werden Ihr Anliegen in kürzester Zeit bearbeiten oder direkt beantworten.

Indi.An Systems GmbH
Flughafenallee 3
28199 Bremen
Deutschland
support@indi-systems.de
Tel + 49 421-989703-30
Fax + 49 421-989703-39