First steps with the PLCcom SDK V.9

Indi.Systems GmbH

Universitätsallee 23
28359 Bremen
Germany
info@indi-systems.de

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

Table of contents

Indi.Systems GmbH
Universitätsallee 23
28359 Bremen
Germany
info@indi-systems.de
Tel + 49 421-989703-30
Fax + 49 421-989703-39

About this document

This document is intended to provide you with information about the provided functionalities. This is not a complete documentation, but a guide to help you getting started.

Further information can be found in

- Code examples within the supplied software package
- Code examples and FAQ on our website http://www.plccom.net/code-examples/plccom/s7.html
- The online help (index.html) within the software package

What provides PLCCom for S7 SDK?

The PLCcom library is a highly performant and optimized component especially designed for Java / .Net-Developer to access data provided by a Siemens PLC in the most comfortable way. With PLCcom you can read and write data from a PLC with ease.

Depending on the package the library consist purely of Java or .Net files. The driver can be integrated directly as a reference so that no API calls are necessary. PLCcom is platform independent and works on 32 as well as 64bit.

Within the scope of supply are various code examples to illustrate the connection between your application and different controllers. These examples can also be used in your own projects.

In its current version, PLCcom is compatible with S7 controllers (200, 300, 400, 1200, 1500, SoftSPS WinAC RTX as well as Logo! 0BA7 / 0BA8) as well as CPUs from other manufacturers (e.g. VIPA 100V/200V/300V/300S, etc.)

More features of the PLCcom S7 library:

- Read and write operation for the following datatypes
  - Raw (Byte or Bool)
  - Bit
  - Byte
  - Word
  - DWord
  - Float
  - INT
  - DINT
  - 16Bit BCD
  - DATE_AND_TIME
  - String
  - S7String
- Read inputs, outputs, data blocks, flags, timer, counter
- **Multiple operations in a single function call. Every function returns a detailed result.**
- Simultaneous access of multiple CPUs
- High performance access. Requests sent to the controller are reduced to the absolute necessary
- Internal functions to read and write certain PLC datatypes with ease
- Supply of the PLCcom dataserver for cyclically read accesses with event controlled notifications on value changes
- Start and stop functionality for the CPU
- Read serial number as well as firmware version
- Read position of key switches
- Read LED info
- Read and write PLC time
- Read system status list

- Read block list
- Read block lengths
- Backup blocks
- Restore blocks
- Restore blocks with changed block number
- Delete blocks
- Read block details like code, generation language, author and more
- Send connection password
- Read diagnostic data
- Auto Connect
- Asynchronous connect
- …and many more

Which programming platforms does PLCcom support?

PLCcom for S7 is available in three versions:

1. .Net version
   The .Net version supports classic .Net Framework programming.
   Furthermore, the delivery package contains a version for .Net-Standard Version 2.0. This component can be used to develop .Net Core, Xamarin, UWP or Unity applications.

2. Windows CE-version
   It supports the development of applications under Windows CE Version 5 or higher.

3. Java version
   The Java version provides developers with a Java component for Java application development, e.g. with Eclipse or Netbeans.
   Since version 9, the development of Android apps is also supported.

Which system components are required for the operation of PLCcom?

The following system components are required for the operation of PLCcom:

- Microsoft .NET Framework 3.5 or higher (.Net version)
- Microsoft .NET Core 2 or higher (.Net version)
- Xamarin.iOS 10.14 or higher (.Net version)
- Xamarin.Mac 3.8 or higher (.Net version)
- Xamarin.Android 8.0 or higher (.Net version)
- Microsoft .NET Compact Framework 3.5 (CE version)
- Java JRE 7 or higher (Java version)
- Android API 14 or higher (Java Version)
- External component Jssc to provide serial ports for Java (Java Version, but not for Android see chapter "Important notes on using PLCcom on Android")

To execute code examples:

- Visual Studio 2010 or higher (.Net-Version)
- Visual Studio 2008 (CE Version)
- NetBeans 8 or 4.5 or higher (Java Version)
- Ecplipse in version Mars or higher (Java Version)

Important notes for using PLCcom on Android

Due to the lack of serial ports, it is not possible to use the following objects on Android:

PLCCom.MPI_Device
PLCCom.PPI_Device

Attempting to create an instance of the above classes ends with an error of type **java.lang.NoClassDefFoundError**

Furthermore, the usage of the PLCCom.FileSystemConnector object must be at least Android API Version 26 or higher, in deviation to the above mentioned information.

Due to the general system conditions under Android, it is assumed that the use of the above-mentioned objects on Android does not matter and the restrictions are negligible.

Passing the license key

Before using the PLCcom library, you must have a license key for the current version. You can either purchase this license key as a time-limited trial key or purchase a license key in advance.

This license key must be passed to the library as follows:

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"
```

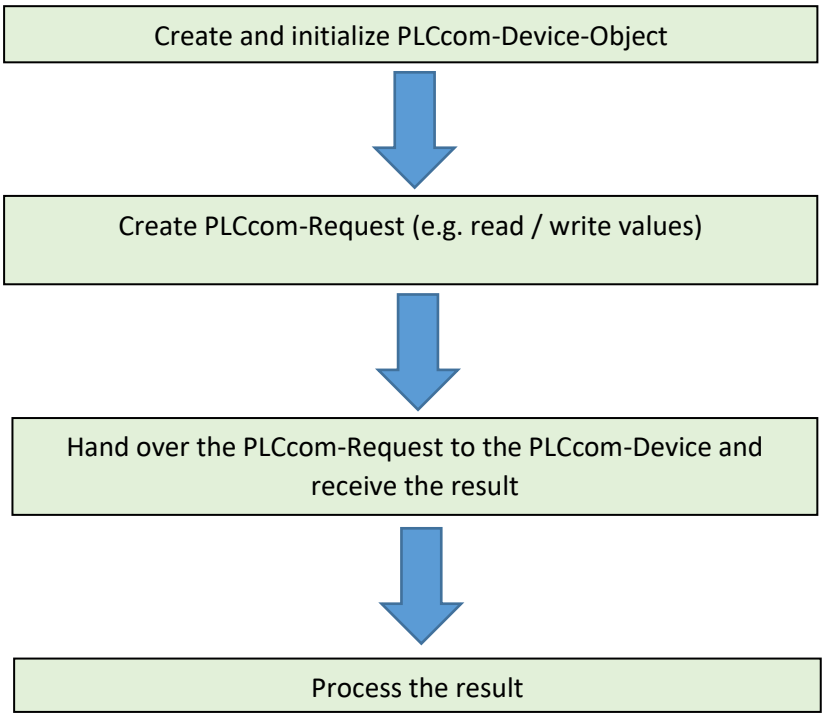
Java

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

Indi.Systems GmbH
Universitätsallee 23
28359 Bremen
Germany
info@indi-systems.de
Tel + 49 421-989703-30
Fax + 49 421-989703-39

First programming steps with PLCcom

Embedding the PLCcom library in your project couldn't be easier. It's done with a few lines of code as shown in this few steps:



Syntax: Create and initialize PLCcom-Device-Object

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);
```

Indi.Systems GmbH
Universitätsallee 23
28359 Bremen
Germany
info@indi-systems.de
Tel + 49 421-989703-30
Fax + 49 421-989703-39

Syntax: Create a simple PLCcom-Request e.g. read data

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
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
```

Indi.Systems GmbH
Universitätsallee 23
28359 Bremen
Germany
info@indi-systems.de
Tel + 49 421-989703-30
Fax + 49 421-989703-39

Syntax: Hand over the request to the PLCcom-Device and receive the result

CSharp

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

Visual Basic

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

Java

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

Syntax: Process the result

```
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.Systems GmbH
Universitätsallee 23
28359 Bremen
Germany
info@indi-systems.de
Tel + 49 421-989703-30
Fax + 49 421-989703-39

Java

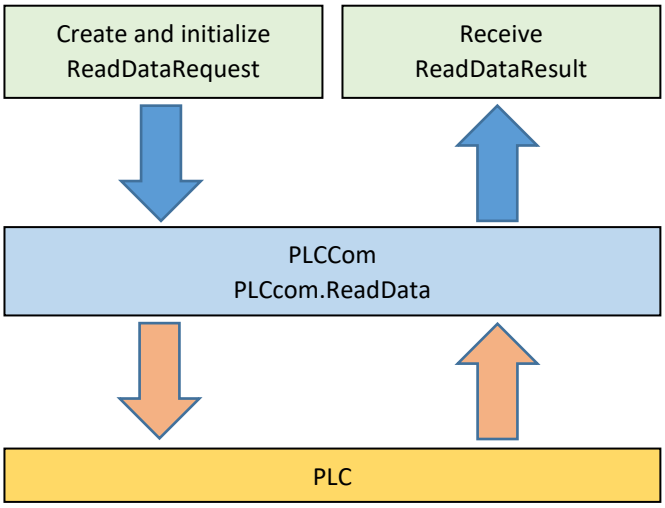
```
//evaluate results
if (res.Quality() == OperationResult.eQuality.GOOD) {
   int Position = 0;
   for (Object item : res.getValues()) {
     System.out.println(item.toString());
   }
}
```

Indi.Systems GmbH
Universitätsallee 23
28359 Bremen
Germany
info@indi-systems.de
Tel + 49 421-989703-30
Fax + 49 421-989703-39

Feature overview

Simple reading data from the PLC

The reading of data is initialized by an ReadDataRequest and executed with the function ReadData().
The result is received as a ReadDataResult object.



Example:

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());
    }
}
```

Visual Basic

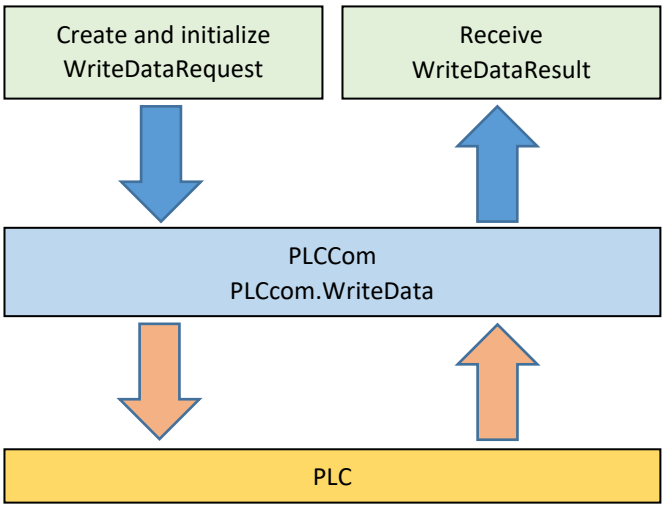
```
Dim Device As PLCcomDevice = New TCP_ISO_Device("192.168.1.2", 0, 2, _

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)

Console.WriteLine((DateTime.Now.ToString() & ": ") + res.Message)

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
```

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());
   }
}
```

Simple writing data to the PLC

The writing of data will be initialized by a WriteDataRequest object and executed by the WriteData() function. The response is a WriteDataResult object.



Example:

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);
   }
```

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());
}
```

Optimized reading and writing of data

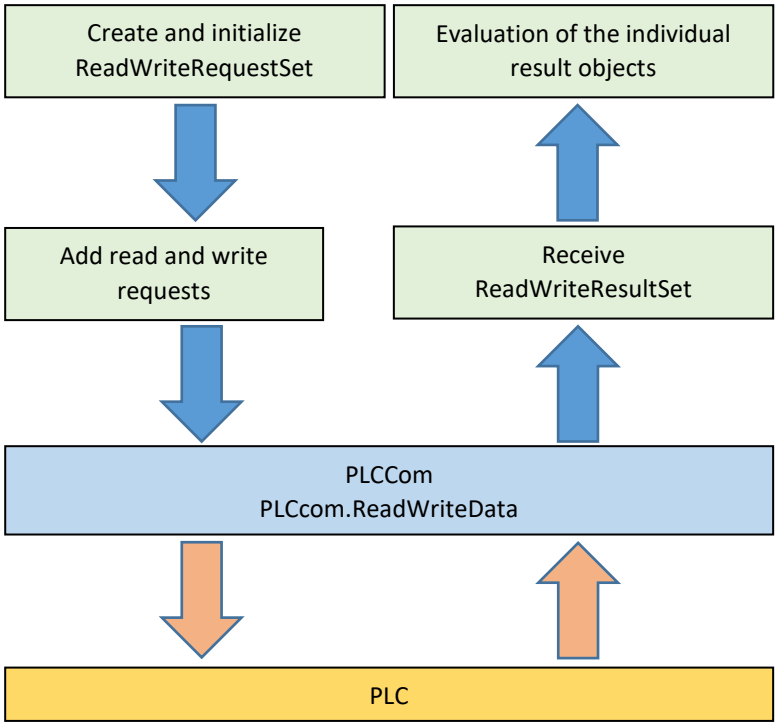
For optimized reading and writing, the read and write accesses in a ReadWriteRequestSet object are grouped together.

The optimized access to the PLC data is initialized via one or more ReadDataRequest and / or WriteDataRequest objects.

These requests are passed to the ReadWriteRequestSet-Object and triggered with the command ReadWriteData and processed optimally on request.

The result of the operation is returned to the developer as ReadWriteResultSet, where the individual ReadDataResult and / or WriteDataResult objects are located for further evaluation.

Function not available for PPI connections, please refer to the function overview on our website www.plccom.net



Optimization options

The ReadWriteRequestSet object has been equipped with various parameterization and optimization options.

The method **RequestSet.SetOperationOrder(eOperationOrder)** can be used to specify whether the write or read accesses are performed first. By default, the data is first written and then read.

The following methods are used to parameterize the data optimization to be performed:

RequestSet.SetReadOptimizationMode(eReadOptimizationMode) and **RequestSet.SetWriteOptimizationMode(eWriteOptimizationMode).** By default, the optimizations are disabled "NONE".

The enum eReadOptimizationMode has the following members:

- NONE:
  No optimization, all read requests are read one after the other. Safe but slow.
- CROSS_AREAS:
  In CROSS_AREAS mode, the read requests are merged across areas. Advantage: fragmented areas (e.g., data across multiple datablocks) can be read and written simultaneously
- COMBINE_AREAS:
  In COMBINE_AREAS mode, read requests from the same areas are combined. Advantage: Fast and high-performance access to data of the same areas (for example, data in the same datablock)
- AUTO:
  PLCcom automatically selects the best optimization method. Only the minimum required PLC read accesses are carried out.
  Only in Expert edition available

The enum eWriteOptimizationMode has the following members:

- NONE:
  No optimization, all read requests are written one after the other. Safe but slow.
- CROSS_AREAS:
  In CROSS_AREAS mode, the read requests are merged across areas. Advantage: fragmented areas (e.g., data across multiple datablocks) can be read and written simultaneously

Example:

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
```

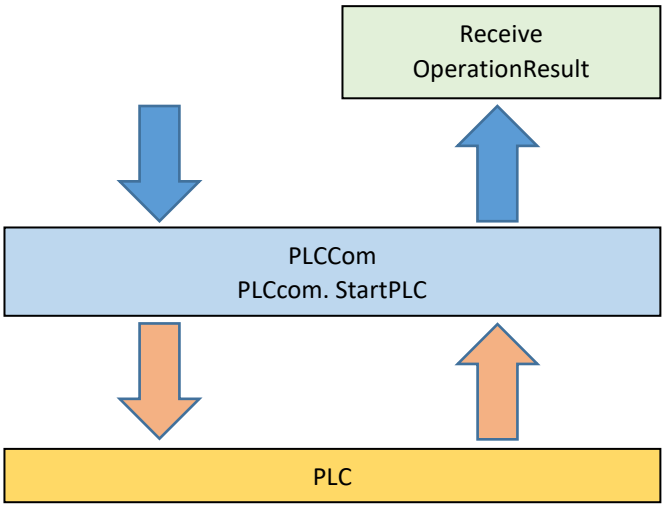
Indi.Systems GmbH
Universitätsallee 23
28359 Bremen
Germany
info@indi-systems.de
Tel + 49 421-989703-30
Fax + 49 421-989703-39

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
}
```

Start PLC

A PLC can be started by calling StartPLC(). An OperationResult object is received as the result.
Not available for all CPUs. Please note the function overview on our website www.plccom.net.



Example:

CSharp

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

Visual Basic

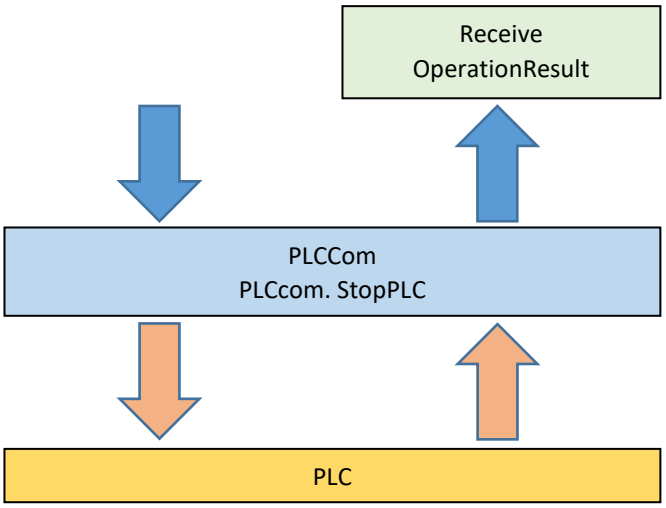
```
Dim res As OperationResult = Device.StartPLC()
```

Java

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

Stop PLC

A PLC can be stopped by calling StopPLC(). An OperationResult object is received as the result.
Not available for all CPUs. Please note the function overview on our website www.plccom.net.



Example:

CSharp

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

Visual Basic

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

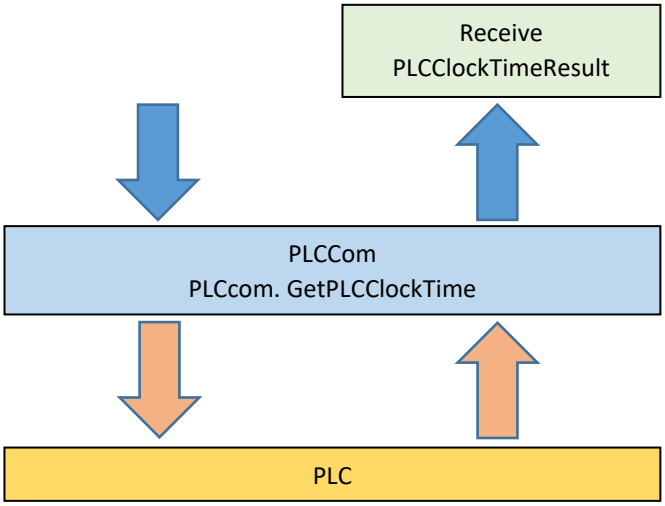
Java

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

Get PLC time

To get the current time of the PLC you can use the function getPLCClockTime(). A PLCClockTimeResult object will be returned.

Not available for all CPUs. Please note the function overview on our website www.plccom.net.



Example:

CSharp

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

Visual Basic

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

Java

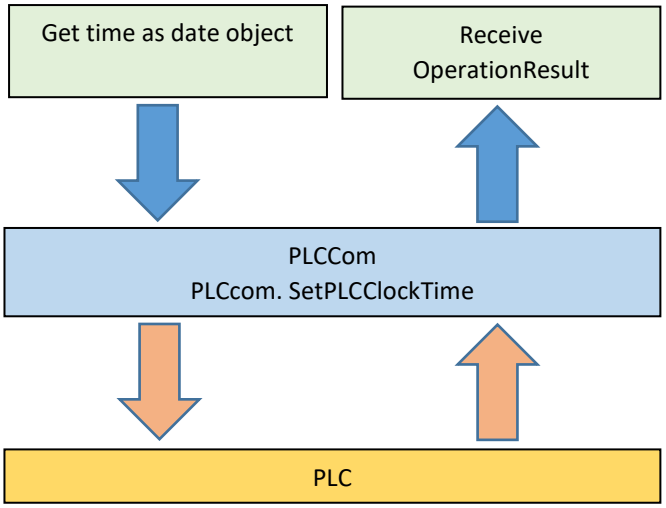
```
PLCClockTimeResult res = Device.getPLCClockTime();
```

Indi.Systems GmbH
Universitätsallee 23
28359 Bremen
Germany
info@indi-systems.de
Tel + 49 421-989703-30
Fax + 49 421-989703-39

Set PLC time

The time of the PLC can be set by using the SetPLCClickTime command. A date object has to be given as a parameter to the function. A PLCClockTimeResult object will be returned as the result.

Not available for all CPUs. Please note the function overview on our website www.plccom.net.



Example:

CSharp

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

Visual Basic

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

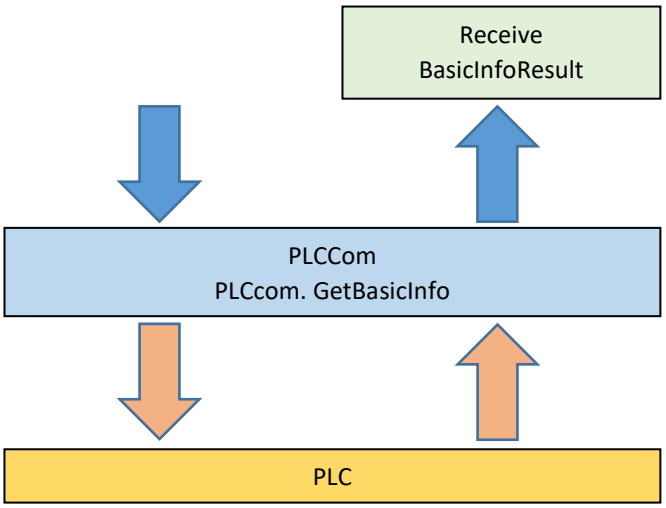
Java

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

Get basic PLC information

With the command GetBasicInfo you can get information like type, module version or firmware version of the CPU. A BasicInfoResult object is returned as the result.

Not available for all CPUs. Please note the function overview on our website www.plccom.net.



Example:

CSharp

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

Visual Basic

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

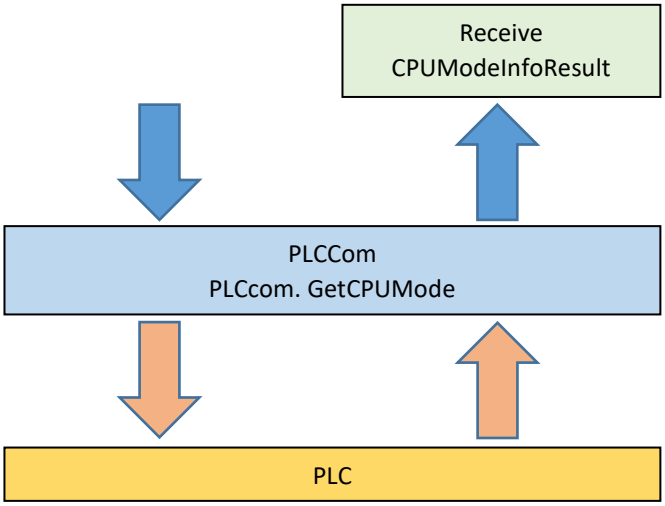
Java

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

Get CPU mode

To get the current mode of the CPU (e.g. run, startup, stop, etc.) you can use the command GetCPUMode. A CPUModeInfoResult object will be returned as the result.

Not available for all CPUs. Please note the function overview on our website www.plccom.net.



Example:

CSharp

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

Visual Basic

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

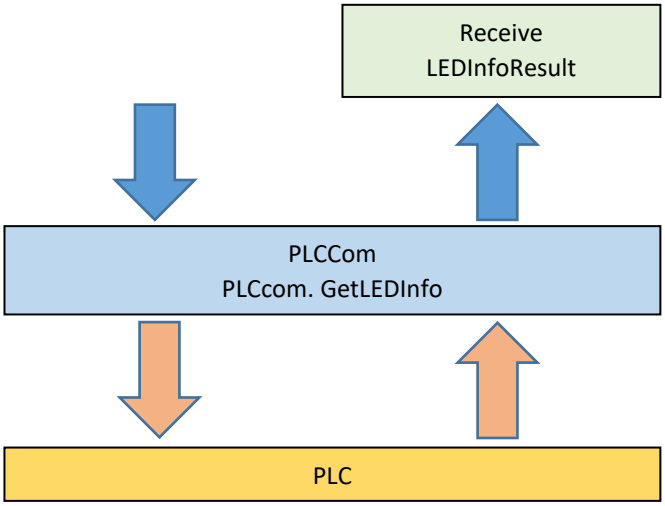
Java

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

Get LED state

To get the current LED state (e.g. on, off, flashing, etc.) you can use the command GetLEDInfo. A LEDInfoResult object will be returned as the result.

Not available for all CPUs. Please note the function overview on our website www.plccom.net.



Example:

CSharp

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

Visual Basic

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

Java

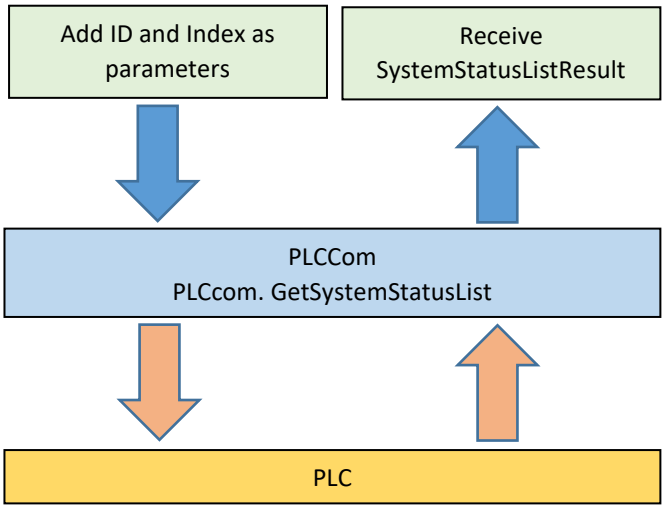
```
LEDInfoResult res = Device.getLEDInfo();
```

Indi.Systems GmbH
Universitätsallee 23
28359 Bremen
Germany
info@indi-systems.de
Tel + 49 421-989703-30
Fax + 49 421-989703-39

Get system status list

The function GetSystemStatusList enables you to access the system status list in the PLC. For further information on the system status list contact the PLC manufacturer or read the corresponding manual. As a result a SystemStatusList object will be returned.

Not available for all CPUs. Please note the function overview on our website www.plccom.net.



Example:

CSharp

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

Visual Basic

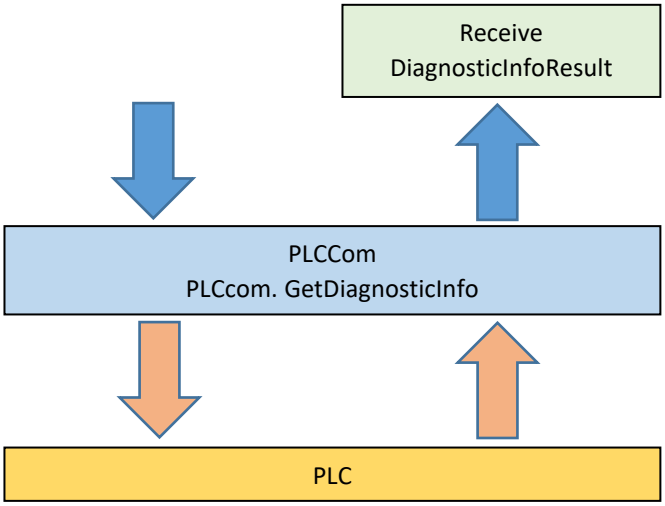
```
Dim res As SystemStatusListResult = Device.GetSystemStatusList(SSL_ID, SSL_Index)
```

Java

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

Get diagnostic info

The command GetDiagnosticInfo serves for requesting the current diagnostic data list from the CPU.
As the result a DiagnosticInfoResult object will be returned.
Not available for all CPUs. Please note the function overview on our website www.plccom.net.



Example:

CSharp

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

Visual Basic

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

Java

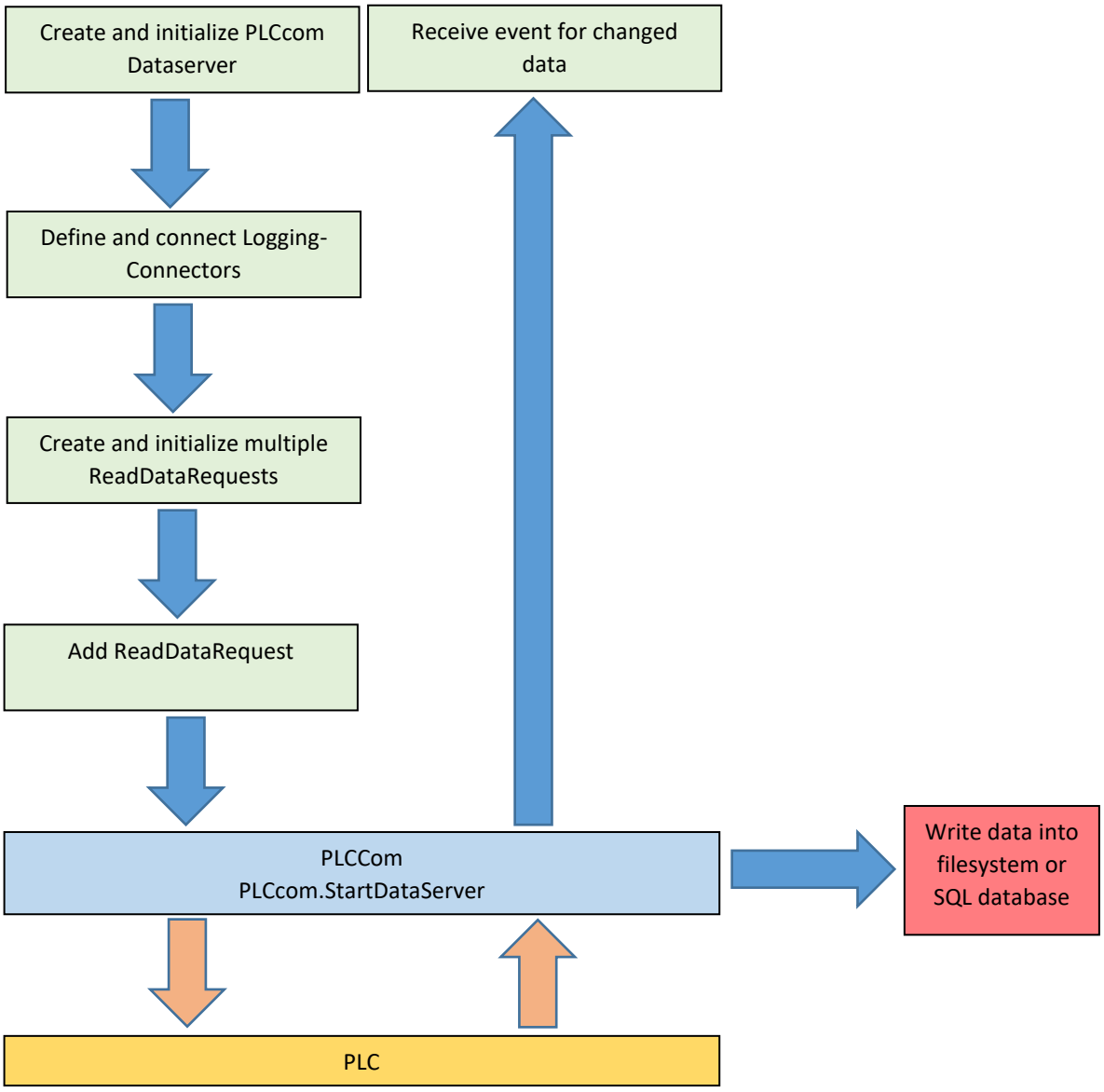
```
DiagnosticInfoResult res = Device.getDiagnosticInfo();
```

Indi.Systems GmbH
Universitätsallee 23
28359 Bremen
Germany
info@indi-systems.de
Tel + 49 421-989703-30
Fax + 49 421-989703-39

PLCcom data server (only in expert version)

With the PLCcom data server you're able to monitor memory ranges on the PLC. Whenever data changes an event is fired to inform the user. The communication between PLCcom and the PLC is optimized in a way that needed requests are reduced to a minimum.

Furthermore it's possible to define connectors for filesystem or SQL database logging. It's also possible to save an image of current variables and values into the filesystem or SQL database. For extended security these data can be stored encrypted. Methods for decryption are enclosed.



Example:

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 Loggingkonnektoren 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()
```

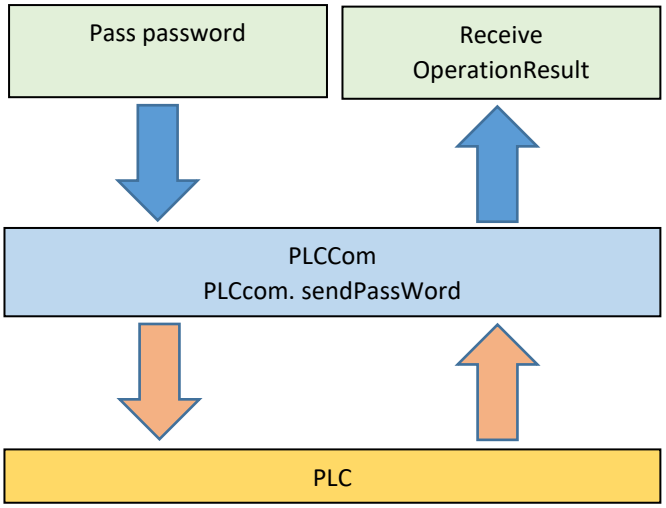
Java

```
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();
```

Send password (only in expert version)

The function sendPassWord allows users to unlock password protected PLCs by passing the password to the function. An OperationResult object will be returned.

Not available for all CPUs. Please note the function overview on our website www.plccom.net.



Example:

CSharp

```
OperationResult res = Device.sendPassWord("PW");
```

Visual Basic

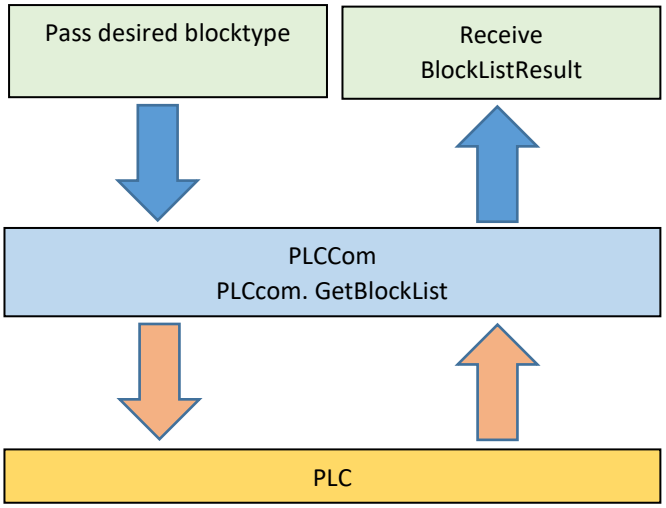
```
Dim res As OperationResult = Device.sendPassWord("PW")
```

Java

```
OperationResult res = Device.sendPassWord("PW");
```

Get object list from PLC (only in expert version)

With the command GetBlockList you're able to get a complete object list, or a list of specific block type (e.g. data block), from the PLC. A BlockListResult object is returned which contains all data.
Not available for all CPUs. Please note the function overview on our website www.plccom.net.



Example:

CSharp

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

Visual Basic

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

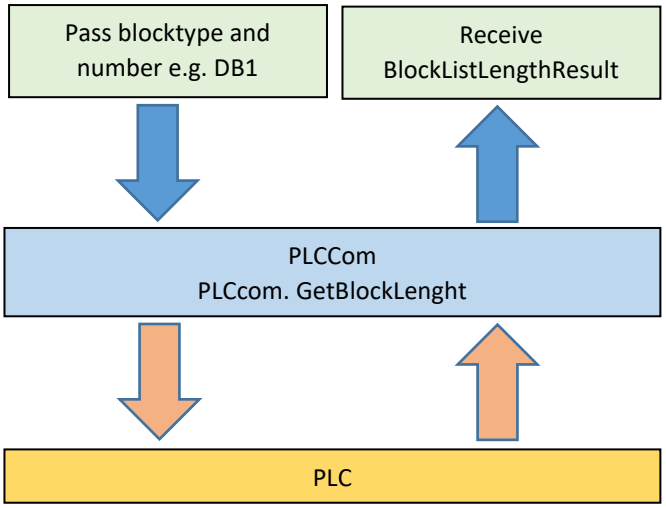
Java

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

Get length of an object (only in expert version)

The command GetBlockLength returns a BlockListLengthResult object containing the length of the requested object from the PLC.

Not available for all CPUs. Please note the function overview on our website www.plccom.net.



Example:

CSharp

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

Visual Basic

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

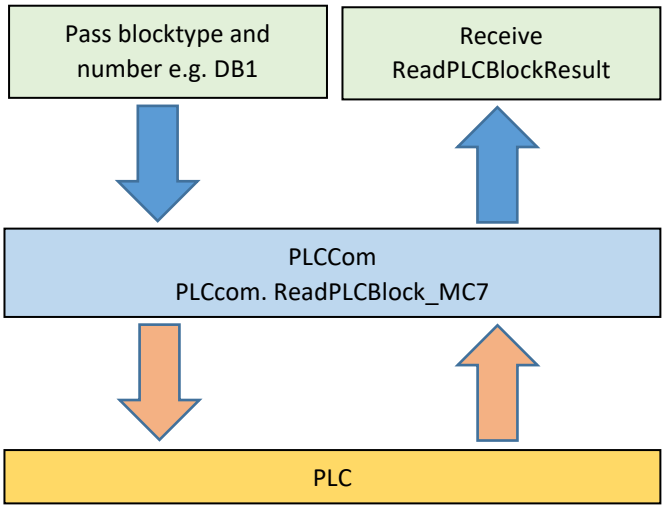
Java

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

Backup an object (only in expert version)

With the function ReadPLCBlock_MC7 the code of a specific block can be accessed and saved in mc7 format. Further block details like code, language, author, etc. are read. A ReadPLCBlockResult object will be returned.

Not available for all CPUs. Please note the function overview on our website www.plccom.net.



Example:

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
```

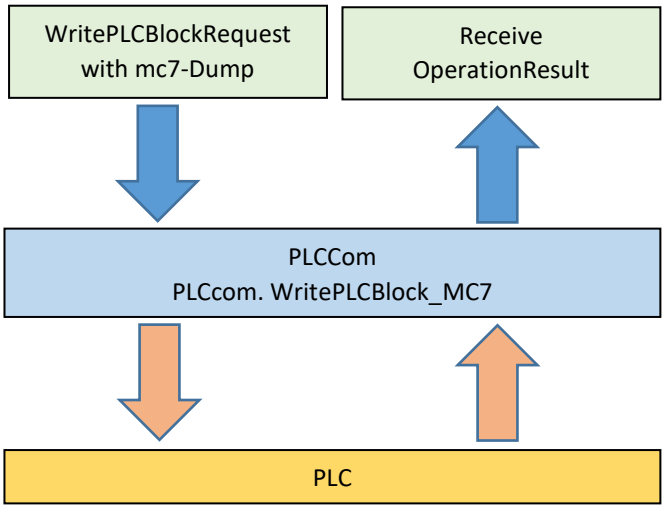
Java

```
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 an object (only in expert version)

Supplemental to the backup function, the function WritePLCBlock_MC7 restores a backup file back into the PLC. It's also possible to restore the data with a changed block number. To pass the data a WritePLCBlockRequest is used. An OperationResult object will be returned.

Not available for all CPUs. Please note the function overview on our website www.plccom.net.



Example:

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
```

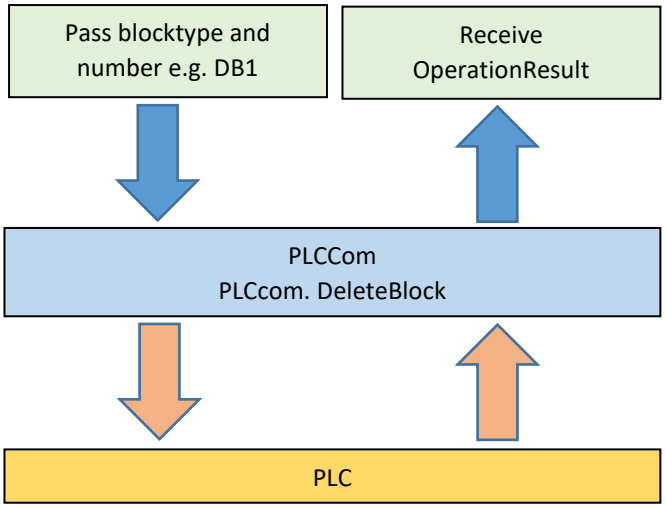
Java

```
// 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);
```

Delete an PLC object (only in expert version)

The function DeleteBlock enables one to delete a specific PLC object. An OperationResult object with additional information will be returned.

Not available for all CPUs. Please note the function overview on our website www.plccom.net.



Example:

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.Systems GmbH
Universitätsallee 23
28359 Bremen
Germany
info@indi-systems.de
Tel + 49 421-989703-30
Fax + 49 421-989703-39

Any questions?

Please write an email to support@indi-systems.de.

We will process your request promptly or respond to you directly.

Indi.Systems GmbH
Universitätsallee 23
28359 Bremen
Germany
info@indi-systems.de
Tel + 49 421-989703-30
Fax + 49 421-989703-39