REA JET

KENNZEICHNUNGSLÖSUNGEN
FÜR DIE INDUSTRIE –
MADE IN GERMANY

User Manual

REA-PLC Interface Protocol

REA JET HR, HR*pro*, CL

For control units with firmware: 3.4 · Document version: 1.17 · Last updated: 14.11.2016

REA JET

REA Elektronik GmbH

Teichwiesenstraße 1

64367 Mühltal, Germany

Tel.: +49 6154 638 0

© 2015 REA Elektronik GmbH, all rights reserved

No part of this user manual may be reproduced in any form whatsoever, or reworked, duplicated or disseminated with electronic systems without first obtaining the written permission of REA Elektronik GmbH.

The information contained in this document may be changed without prior notice being given.

Table of contents

1 Document structure

The manual is subdivided into chapters that discuss separate topics. If additional information is available on a procedure or a task, then you will find a cross-reference to the chapter (plus page number) at the corresponding place, to help you quickly locate this supplementary information.

Great care has been taken in writing and reviewing this manual. If you can suggest additional information or improvements, we would be grateful for your valued opinion. We would ask you to forward your suggestions by email, referencing this user manual, to info@rea.de. Thank you!

1.1 Technical terms used

The technical terms are sorted alphabetically in the table.

Technical term	Description
ASCII	The initials stand for "American Standard Code for Information Interchange". In this specification, individual characters are defined from a numerical value of 0 (0x00) to 255 (0xFF).
Byte	A byte consists of 8 bits, where a bit is the smallest unit of storage used by a computer. A bit can assume the state of 0 (false) or 1 (true). A byte can thus represent a numerical value of 0 to 255.
7-bit	From a byte that has 8 bits, only the first 7 bits are evaluated. A byte with 7 bits can thus represent a numerical value of 0 to 127.
DHCP	The initials stand for "Dynamic Host Configuration Protocol". This means that the IP address in the control unit is issued dynamically by the "host" and can therefore change if the control unit is restarted (switched off/on).
IP address	The IP address defines the address at which the control unit can be reached in the network. Each device has its own, unique IP address in the network.
Labelling system	A labelling system consists of at least one control unit and a write head.
Proxy	An intermediary in networks

Technical term	Description
RS-232	A very widely-used serial interface protocol. The maximum cable length is dependent on the transfer speed, which is specified in baud (bit/s).
RS-422	A serial interface protocol that, unlike RS-232, is much more fault-resistant and is therefore suitable for use over greater distances.
Samba protocol	An interface protocol for file transfer in TCP/IP networks.
String	A string is a container that can hold an arbitrary character string. It is typically composed of multiple bytes. A string contains the data content or instructions that are used for communication.
TCP/IP	Stands for "Transmission Control Protocol / Internet Protocol" and is now an established standard. TCP/IP is used by DHCP, FTP, the Samba protocol, and much, much more.

Table 1: Technical terms

1.2 What is a byte?

Since this documentation talks about bytes a lot, here is a little bit of background information.

A byte consists of 8 bits, i.e. a byte can hold numerical values from 0 to 255 if the underlying system being used is the decimal (DEC) number system. The decimal number system is not used for interface protocols, however: instead, the hexadecimal (HEX) number system is typically used. In this system, a byte can hold values from 0x00 to 0xFF, where 0x is a code indicating use of the hexadecimal number system. This is indicated in the rest of this document by the heading "Hex format".

In the following documentation, the instructions and data content are not given using the hexadecimal notation, but in ASCII format, since this is the simplest human-readable format. To avoid having to convert between formats when troubleshooting, hexadecimal values are also given with the ASCII, so as to make it easier to track down errors during data communications.

All in all, there are three ways of notating one and the same entity, and each notation has its benefits and drawbacks.

Example:

A string with nine bytes in hex format, with the following content:

```
00 02 03 0A 20 30 39 41 7A
```

The same string in ASCII, also with nine characters:

```
[NUL] [STX] [ETX] [LF] [SP] 0 9 A z
```

And the same string, this time in decimal format:

```
0 2 3 10 32 48 57 64 122
```

In hex notation, the alphanumeric symbols (A to F) can be notated either using capitals or lower-case letters.

1.2.1 *Valid content for a byte in this PLC protocol*

Although a byte can hold any one of 255 (0xFF) values, not all data content is actually permitted by this protocol. Accordingly, a byte can utilise only the following characters for data communications:

ASCII format:
`[SP] to ~`

Hex format:
`20 to 7E`

For establishing a connection (see chap. **Fehler! Verweisquelle konnte nicht gefunden werden.** on page **Fehler! Textmarke nicht definiert.**), only the following control characters are valid:

ASCII format:
`[STX], [ETX], [LF]`

Hex format:
`02, 03, 0A`

The semicolon character (;) is used in this protocol as a separator. Excepting its use in object content, it must not be used for any other purpose.

ASCII format:
`;`

Hex format:
`3B`

The background for this restriction is that the characters communicated must always be legible and any data communication errors must be found as rapidly as possible.

In the hexadecimal system, characters A to F are generally written as capitals. In practice, the characters can also be written in lower-case: this is supported by many protocols, including the REA PLC protocol. Accordingly, this document may specify an "E" in ASCII format but may use either a "45" or a "65" for the representation in hex format.

1.2.2 *Representation of string length in bytes*

This protocol requires the lengths of user data to be calculated. For an example, please see chap. 2.3.4 on page 13.

In this example, the length of the string is 26 characters. The number 26 is encoded as 0x1A in hex. As described in chapter 1.2.1 on page 6, a byte cannot contain the value 0x1A. Accordingly, these size specifications are split into (two) bytes, as follows:

ASCII format:

```
1A
```

Hex format:

```
31 61
```

The advantage of this method is that the data is always legible in ASCII format. This also affects counters, e.g. the "ID" counter, which increments as follows:

ASCII format	Hex format
00000001	30 30 30 30 30 30 30 31
00000002	30 30 30 30 30 30 30 32
00000003	30 30 30 30 30 30 30 33
00000004	30 30 30 30 30 30 30 34
00000005	30 30 30 30 30 30 30 35
00000006	30 30 30 30 30 30 30 36
00000007	30 30 30 30 30 30 30 37
00000008	30 30 30 30 30 30 30 38
00000009	30 30 30 30 30 30 30 39
0000000a	30 30 30 30 30 30 30 61
0000000b	30 30 30 30 30 30 30 62
0000000c	30 30 30 30 30 30 30 63

ASCII format	Hex format
0000000d	30 30 30 30 30 30 30 64
0000000e	30 30 30 30 30 30 30 65
0000000f	30 30 30 30 30 30 30 66
00000010	30 30 30 30 30 30 31 30
00000011	30 30 30 30 30 30 31 31
00000012	30 30 30 30 30 30 31 32
00000013	30 30 30 30 30 30 31 33

2 REA-PLC

REA-PLC is the name for an interface protocol between a server (labelling system) and a client (such as a PLC control unit) that has been specially designed for PLC systems. The protocol developed by **REA-Elektronik GmbH** is supported by the following labelling systems:

- **REA JET HR**
- **REA JET HR *pro***
- **REA JET CL**

All labelling systems use TCP/IP to communicate with the client. An interface converter (Interfaces) can also be used to connect to RS-422, RS-485 or RS-232 interfaces.

This protocol has been specially developed for communications involving the abovementioned labelling systems and PLC systems. The protocol has the following features:

- Support for the five most important commands.
- No automatic, unrequested data communications.
- Answer length is always the same for the five commands implemented. The standard answer also contains the status information for the control unit and its write head(s).

From a communications architecture perspective, the labelling system is the server and the PLC system/PC is the client. One server can have multiple clients – but not vice versa!

2.1 Connection setup between client and server

Before instructions can be sent to the server, the client first needs to establish a connection to the server. Although this connection is a 1:1 connection over the TCP/IP layer, multiple simultaneous connections are actually feasible, since a server can receive TCP/IP packets from more than one client.

To establish a connection, you will require

- An IP address
- A port number

The IP address is available from your labelling system: to find out where this address is documented, please consult the operating instructions provided with your labelling system. The address can usually be accessed with <**F5**> in the Basic View.

The labelling system is preconfigured (factory settings) and uses DHCP: this means the IP address of your labelling system changes dynamically every time the labelling system is switched off and on.

The port number depends on its application. **REA-PLC** supports two separate variants, each of which can be selected with the appropriate port number.

2.1.1 *Interface protocol on port 22169 (EOT), from firmware 1.83*

This interface protocol does not use a handshake and adds a separator character to the default answer – the control character [EOT] (0x04) is used for this purpose. This also enables communication that works like PLC but requires a terminator character to recognise the end of the string – as with some camera systems, for example.

2.1.2 *Interface protocol on port 22170, from firmware 1.70*

The interface protocol is length-based, i.e. strings always have a fixed length and parameters passed are ignored if these also pass string lengths in the parameter set.

2.2 Interface protocol description

An instruction has the following basic structure and contains at least 18 bytes.

```
<Instruction><ID><Length><Parameter>]
```

	Length	Description
Instruction	4 bytes	See chap. 2.3 on page 11
ID	8 bytes	An arbitrary value in the range: 1 to FFFFFFFF, see chap. 1.2 on page 5.
Length	6 bytes	Total number of characters in the following parameter.
Parameter	{} bytes	Depends on instruction

If an instruction is sent successfully to the server, the server automatically sends an answer (response) with this ID and the unmodified instruction code back to the client, see chap. 2.4 on page 17.

2.3 Instructions

Instructions are commands sent by the client to the server, so as to cause the server to execute an action.

For a precise description of instructions, and the conditions under which the instructions can be executed, please consult the documentation provided with your labelling system. As a rule, instructions are sent together with their associated, specific parameters. These parameters are described in detail in the following sections.

2.3.1 Assign print job

This instruction is used to assign a print job to the labelling system. A print job typically consists of system settings and the content to be printed.

General structure of the instruction:

```
<Instruction><ID><Length><Job ID><File name>
```

Structure	Length	Description
Instruction:	4 bytes	0001 ≡ The instruction for "Set Job"
ID	8 bytes	An arbitrary value, see chap. 2.2 on page 11
Length:	6 bytes	Total count of the following parameters. Here, the length is composed of **Job ID** and **`File name`** (**Length** = **Job ID** + **`File name`**)
Job ID:	1 byte	0 ≡ Job 1 Currently, only Job 1 is supported
`File name:`	{} bytes	Name of the job file, which must already be present on the server. Note: case-sensitive!

Example – assigning a print job with the name "demojob_1ph.job" to job no. 1

<u>ASCII format:</u>

```
000100000001000010Odemojob_1ph.job
```

<u>Hex format:</u>

```
30 30 30 31 30 30 30 30 30 30 30 31 30 30 30 30 31
30 30 64 65 6D 6F 6A 6F 62 5F 31 70 68 2E 6A 6F 62
```

2.3.2 *Start print job*

The control unit is ready to execute the print job only once the print job has been started. The instruction is identical to the <**Start**> key on the control unit.

General structure of the instruction:

```
<Instruction><ID><Length><Job ID>
```

Structure	Length	Description
Instruction:	4 bytes	0002 ≡ The instruction for "Start Job"
ID	8 bytes	An arbitrary value, see chap. 2.2 on page 11
Length:	6 bytes	Total number of characters in the following parameter. Here, this is only the **Job ID** – the length is therefore always 1.
Job ID:	1 byte	0 ≡ Job 1 Currently, only Job 1 is supported

Example – starting print job no. 1:

ASCII format:

```
000200000001000001O
```

Hex format:

```
30 30 30 32 30 30 30 30 30 30 30 31 30 30 30 30 30
31 30
```

2.3.3 *Stop print job*

This instruction is the opposite of "Start print job" – it stops the job. The instruction is identical to the <**Stop**> key on the control unit.

General structure of the instruction:

```
<Instruction><ID><Length><Job ID>
```

Structure	Length	Description
Instruction:	4 bytes	3 ≡ The instruction for "Stop Job"
ID	8 bytes	An arbitrary value, see chap. 2.2 on page 11
Length:	6 bytes	Total number of characters in the following parameter. Here, **Job ID** – the length is therefore always 1.
Job ID:	1 byte	0 ≡ Job 1 Currently, only Job 1 is supported

Example – stopping print job no. 1:

ASCII format:

```
0003000000010000010
```

Hex format:

```
30 30 30 33 30 30 30 30 30 30 30 31 30 30 30 30 30
31 30
```

2.3.4 *Overwrite object contents*

This instruction can be used to overwrite or modify object contents.

General structure of the instruction:

```
<Instruction><ID><Length>
<ID length>
<Job ID>;<Group name>;<Object name>;<Content name>
<Content length>
<Content>
…

<ID length>
<Job ID>;<Group name>;<Object name>;<Content name>
<Content length>
<Content>
```

The group name used, together with the object names and content names used, are already preconfigured in the system settings or the creation of the label itself. For details, see the "**REA JET LabelCreator**" manual.

Structure	Length	Description
Instruction:	4 bytes	0004 ≡ The instruction for "Set Label Contents"
ID	8 bytes	An arbitrary value, see chap. 2.2 on page 11
Length:	6 bytes	Total number of characters in the following parameter.
ID length:	4 bytes	Number of characters for: `Job ID + Group name` + Object name + `Content name` + 3 (semicolon as separator)
Job ID:	1 byte	0 ≡ Job 1 Currently, only Job 1 is supported
Separator:	1 byte	Always semicolon
Group name:	{} bytes	Name of the group Note: case-sensitive!
Separator:	1 byte	Always semicolon
Object name:	{} bytes	Name of the object Note: case-sensitive!
Separator:	1 byte	Always semicolon
Content name:	{} bytes	Name of the content Note: case-sensitive!
Content length:	4 bytes	Number of content characters
Content:	{} bytes	The actual new content
…		
ID length:	4 bytes	Number of characters for:

Structure	Length	Description
		Job ID + **Group name** + **Object name** + **Content name** + 3 (semicolon as separator)
Job ID:	1 byte	0 ≡ Job 1 Currently, only Job 1 is supported
Separator:	1 byte	Always semicolon
Group name:	{} bytes	Name of the group Note: case-sensitive!
Separator:	1 byte	Always semicolon
Object name:	{} bytes	Name of the object Note: case-sensitive!
Separator:	1 byte	Always semicolon
Content name:	{} bytes	Name of the content Note: case-sensitive!
Content length:	4 bytes	Number of content characters
Content:	{} bytes	The actual new content

Example – overwriting two object contents with "REA Elektronik GmbH" and "?":

ASCII format:

```
0004000000001000062
001a0;Front;Test-Text_1;Text_1
0013REA Elektronik GmbH
00240;Front;Test-Text_2;Exclamation-mark
0001?
```

Hex format:

```
30 30 30 34 30 30 30 30 30 30 30 31 30 30 30 30 36
32 30 30 31 61 30 3B 46 72 6F 6E 74 3B 54 65 73 74
2D 54 65 78 74 5F 31 3B 54 65 78 74 5F 31 30 30 31
33 52 45 41 2D 45 6C 65 6B 74 72 6F 6E 69 6B 20 47
6D 62 48 30 30 32 34 30 3B 46 72 6F 6E 74 3B 54 65
73 74 2D 54 65 78 74 5F 32 3B 45 78 63 6C 61 6D 61
74 69 6F 6E 2D 6D 61 72 6B 30 30 30 31 21
```

Auxiliary calculation for ID length:

Job ID:	0	1 character
Separator	Semicolon	1 character
Group name:	Front	5 characters
Separator	Semicolon	1 character
Object name:	Test-Text_1	11 characters
Separator	Semicolon	1 character
Content name:	Text_1	6 characters
===		
Total:		26 characters ≡ (hex: **1A**)

2.3.5 *Overwrite object properties*

From firmware 3.20

This instruction can be used to overwrite or modify object properties. The following object properties are possible but are not available for all objects:

Object properties	Unit	Description
Position/X@value	mm	Numerical decimal, e.g. "1.5", "6"
Position/Y@value	mm	Numerical decimal, e.g. "1.5", "2.346"
Position/Z@transparency	[bool]	Boolean value: – True := "True" or "true" – False := {}
Size/Width@value	mm	Width of the object Numerical decimal, e.g. "20", "40.123"
Size/Height@value	mm	Numerical decimal, e.g. "12", "6.34"
HiddenCount@value	[]	Numerical, positive integer value with number of print runs in which the object is hidden: -1 := The object is always hidden 0 := The object is always shown N := Number of print runs until the object will be shown again, e.g. "14", "1", "2"
Inverted@value	[bool]	Boolean value: – True := "True" or "true" – False := {}
Rotation@value	°	Numerical decimal specifying the clockwise angle of rotation, from 0 to 360 degrees (for HR, in 90-degree steps only), e.g. "90", "37.57"
Font/NameEmphasis@value	[string]	A character string with the valid name and font style of an installed system font, separated by a forward slash, e.g. "FreeSans/Medium"
Content@value	[string]	A character string of arbitrary length and content, ASCII/ANSI-coded (7-bit characters), e.g. "Hello World"

Note: the object properties are case-sensitive!

General structure of the instruction:

```
<Instruction><ID><Length>
<ID length>
<Job ID>;
<Group name>;<Object name>;<Content name>;<Object
property>
<Content length>
<Content>
…
<ID length>
<Job ID>;<Group name>;<Object name>;<Content
name>;<Object property>
<Content length>
<Content>
```

Specification of the content name is not required for any object property except "Content@value".

The group name used, together with the object names and content names used, are already preconfigured in the system settings or the creation of the label itself. For details, see the "**REA JET LabelCreator**" manual.

Structure	Length	Description
Instruction:	4 bytes	5 ≡ The instruction for "Set label object" (hex: 0005)
ID	8 bytes	An arbitrary value, see chap. 2.2 on page 11
Length:	6 bytes	Total length of the data string, without **Instruction,** ID and the **Length** itself, but including the semicolon.
ID length:	4 bytes	Number of characters for: `Job ID + Group name` + object name + `Content name + Property` + 4 (semicolon as separator)
Job ID:	1 byte	0 ≡ Job 1 Currently, only Job 1 is supported
Separator:	1 byte	Always semicolon
Group name:	{} bytes	Name of the group Note: case-sensitive!
Separator:	1 byte	Always semicolon
Object name:	{} bytes	Name of the object Note: case-sensitive!
Separator:	1 byte	Always semicolon
`Content name:`	{} bytes	Name of the content Note: case-sensitive!

Structure	Length	Description
Separator:	1 byte	Always semicolon
Object property:	{} bytes	Name of the property – see above Note: case-sensitive!
Content length:	4 bytes	Number of content characters
Content:	{} bytes	The actual new content
…		
ID length:	4 bytes	Number of characters for: `Job ID + Group name` + Object name + `Content name + Property` + 4 (semicolon as separator)
Job ID:	1 byte	0 ≡ Job 1 Currently, only Job 1 is supported
Separator	1 byte	Always semicolon
Group name:	{} bytes	Name of the group Note: case-sensitive!
Separator:	1 byte	Always semicolon
Object name:	{} bytes	Name of the object Note: case-sensitive!
Separator:	1 byte	Always semicolon
`Content name:`	{} bytes	Name of the content Note: case-sensitive!
Separator:	1 byte	Always semicolon
Object property:	{} bytes	Name of the property – see above Note: case-sensitive!
Content length:	4 bytes	Number of content characters
Content:	{} bytes	The actual new content

Example – changing an object property ("Position/X@value") to 10 mm:

ASCII format:

`00050000000100002f00250`;`Front`;`Test-Text_1`;;
Position/X@value`000220`

Hex format:

```
30 30 30 35 30 30 30 30 30 30 30 31 30 30 30 30 32
66 30 30 32 35 30 3B 46 72 6F 6E 74 3B 54 65 73 74
2D 54 65 78 74 5F 31 3B 3B 50 6F 73 69 74 69 6F 6E
2F 58 40 76 61 6C 75 65 30 30 30 32 32 30
```

Example – overwriting an object content ("Content@value") with REA-Elektronik GmbH:

<u>ASCII format:</u>

```
00050000000100004300280;Front;Test-Text_1;Text_1;Content@value0013REA-Elektronik GmbH
```

<u>Hex format:</u>

```
30 30 30 35 30 30 30 30 30 30 30 31 30 30 30 30 34
33 30 30 32 38 30 3B 46 72 6F 6E 74 3B 54 65 73 74
2D 54 65 78 74 5F 31 3B 54 65 78 74 5F 31 3B 43 6F
6E 74 65 6E 74 40 76 61 6C 75 65 30 30 31 33 52 45
41 2D 45 6C 65 6B 74 72 6F 6E 69 6B 20 47 6D 62 48
```

2.4 Answers from server (response)

The interface protocol on port 22170 has an answer length of 64 bytes. With the interface protocol on port 22169, the data end character [EOT] (hex: 4) will be added, thus making the response 65 bytes long in total.

The first 64 bytes in all protocols have the following meaning:

Structure	Length	Description
Instruction	4 bytes	The instruction that was executed
ID	8 bytes	The same ID that was sent with the instruction to the server
Error code	8 bytes	See chap. 2.4.4 on page 22
Device status	4 bytes	Currently, the value is always null (hex: 0000)
Job status	8 bytes	Print job status, see chap. 2.4.1 on page 21
Status field	32 bytes	Status information, see chap. 2.4.3 on page 21

2.4.1 Instruction

The instruction received (assign/start/stop, etc. print job) – see chap. 2.3 from page 11 – is returned again by the server at this point. If the server does not recognise the instruction received, the server returns "FFFF" instead of the instruction code received.

2.4.2 Job status

The following table returns the corresponding job status.

Job status	Description
0000xxxx	No job assigned and no print release
0001xxxx	Job is assigned, but no print release
0002xxxx	Error! Print release but no job assigned – this state should be impossible!
0003xxxx	Job is assigned and has been released for printing

2.4.3 Status field

Interpretation of the status field depends on the server (control unit) in question.

2.4.3.1 Status field: REA JET HR and REA JET HR *pro*

For the **REA JET HR** and the **REA JET HR *pro***, the corresponding ink levels and cartridge status details are returned in the status field. Information about ink levels can only be returned by cartridges that have been fitted with an integrated chip. The full data word is always returned, regardless of the labelling system in use and the current population. This ensures a systematic, consistent approach at all times.

Structure	Length	Description
Cartridge 1	4 bytes	See cartridge status table
Cartridge 1	4 bytes	Contains the ink level in ml (millilitres)
Cartridge 2	4 bytes	See cartridge status table
Cartridge 2	4 bytes	Contains the ink level in ml (millilitres)
Cartridge 3	4 bytes	See cartridge status table
Cartridge 3	4 bytes	Contains the ink level in ml (millilitres)
Cartridge 4	4 bytes	See cartridge status table
Cartridge 4	4 bytes	Contains the ink level in ml (millilitres)

Cartridge status	Description
Bit 0:	Cartridge is inserted
Bit 1:	Cartridge is empty
Bit 2:	Cartridge temperature too high
Bit 3:	Ink level below target level
Bit 4:	Cartridge temperature above target level

2.4.3.2 Status field: REA JET CL

The status field is not currently specified for the **REA JET CL**.

2.4.4 Error codes

Currently, error codes are available in English only.

Error code	Description
0000xxxx	No error
0001xxxx	Print job not started
0002xxxx	Unknown severe error

Error code	Description
0003xxxx	Unknown error
0004xxxx	Invalid parameters
0005xxxx	Fatal error, device will restart
0006xxxx	A memory exception occurred
0007xxxx	Not supported
000A01F4	File not found
000B01F4	Cannot open file
000A0065	Invalid label tag
00140065	Label used resource missing
00190065	Label used bitmap missing
001E0065	Label used font missing
005A0065	Access to label or label tag not granted
00620065	Object content set without any changes
00630065	Object content set produces an stop condition/request for printing
000A0066	You cannot do some actions while running a job
000B0066	You cannot do some actions while NOT running a job
000C0066	There is no active/assigned job
000D0066	Job contains no group
000E0066	Job does not exist
000F0066	Group contains no label
00100066	Tried action on not assigned group
00110066	Invalid ink information
00120066	Print head not ready
00130066	Shaft encoder not configured
00140066	Prerendering at activation failed (probably a broken label)
00150066	An entity of the printing system is active and cannot be (re)parameterised
00160066R	rendering failed, no label-image from renderer available
00170066	You cannot do some actions while job is purging
00180066	Not all printheads/cartridges found are included in a job
00190066	Cannot purge with double existing printheads/cartridges
001A0066	Cannot purge without any printheads/cartridges
001B0066	Cannot purge printhead/cartridge <id> in a running job

Error code	Description
001C0066	Cannot purge not locked or not connected cartridge <id>
001D0066	Group does not exist
001E0066	Group must not be active to perform
001F0066	Group has to be active to perform
00200066	No buffer to store the image data for group
00210066	Group has to contain at least one printhead
00220066	Desired hardware not supported
00230066	Group must not contain printheads with identical IDs
00240066	Group must not be executing purge to perform
00250066	Group/job cannot be started within external stopped condition
00320066	Type of ink for spitting doesn't match with settings
000A0067	Common error in parsing XML
000B0067	Error in XML syntax
000C0067	XML document doesn't contain valid root node
000F0067	XML document invalid or missing job node
00100067	XML document job version not supported
00130067	File of XML document for label not found
00140067	XML document invalid or missing label node
00150067	XML document label version not supported
00280067	Logic error in XML tag
00290067	Error in XML tag label
002A0067	Error in XML tag layout (of label)
002C0067	Error in XML tag object (of label)
002D0067	Error in XML tag renderer (of label)
002E0067	XML tag renderer type (of label) not supported
002F0067	Error in XML tag content (of label object)
00300067	XML tag content type (of label object) not supported
00310067	XML tag renderer type needs content
00360067	XML tag referenced content cannot be found
00460067	Unknown feature
000A0068	Rendering result image failed
000B0068	Cannot create or start render thread
00140068	Invalid size of destination image

Error code	Description
001E0068	Cannot create renderer
001F0068	Cannot create renderer for text out
00230068	Cannot create renderer for images
00240068	Cannot create renderer for barcodes
00280068	Cannot create renderer for graphical objects
00320068	Error in initialise Freetype library
003C0068	Error in initialise TBarcode library
003E0068	Cannot render barcode object
00500068	Render failed on invalid format of source image
00520068	Cannot render bitmap object
00530068	Cannot render image because of expired print count
005A0068	Render failed on invalid shift code
006E0068	Code list for is containing insufficient entries
000A00C8	Label data for FPGA too late
000B00C8	Pulses from shaft encoder too fast
000C00C8	Unsteady pulses from shaft encoder
001400C8	Invalid shaft encoder parameter(s)
001E00C8	Hardware subsystem missing
002800C8	Number of layers exceeded / not supported
003200C8	Invalid FPGA configuration
003C00C8	Failed to copy image to FPGA
004600C8	Logical AND-operation with edges not allowed
005000C8	If command updatefonts fails
000B012C	Print head not connected
000C012C	Print head is not locked
000D012C	Initialisation of print head failed
000E012C	Print head contains no cartridge
000F012C	No ink left in cartridge
0010012C	Print head temperature too high
000A0136	Print cartridge has no chip
000B0136	Cartridge chip communication failure

Error code	Description
000A0190	Event subscription failed
00140190	XML document REA-PI version not supported
00150190	Version already selected

3 Software for test and evaluation purposes

REA-Elektronik GmbH provides PC software for the purposes of testing the PLC interface. This gives users the opportunity to familiarise themselves with the interface protocol more quickly.

3.1 REA-PLC Tester, from version: 1.14.00

REA-PLC Tester is an application that is part of the control unit's standard scope of delivery: it is included as a compressed archive file on the enclosed data medium.

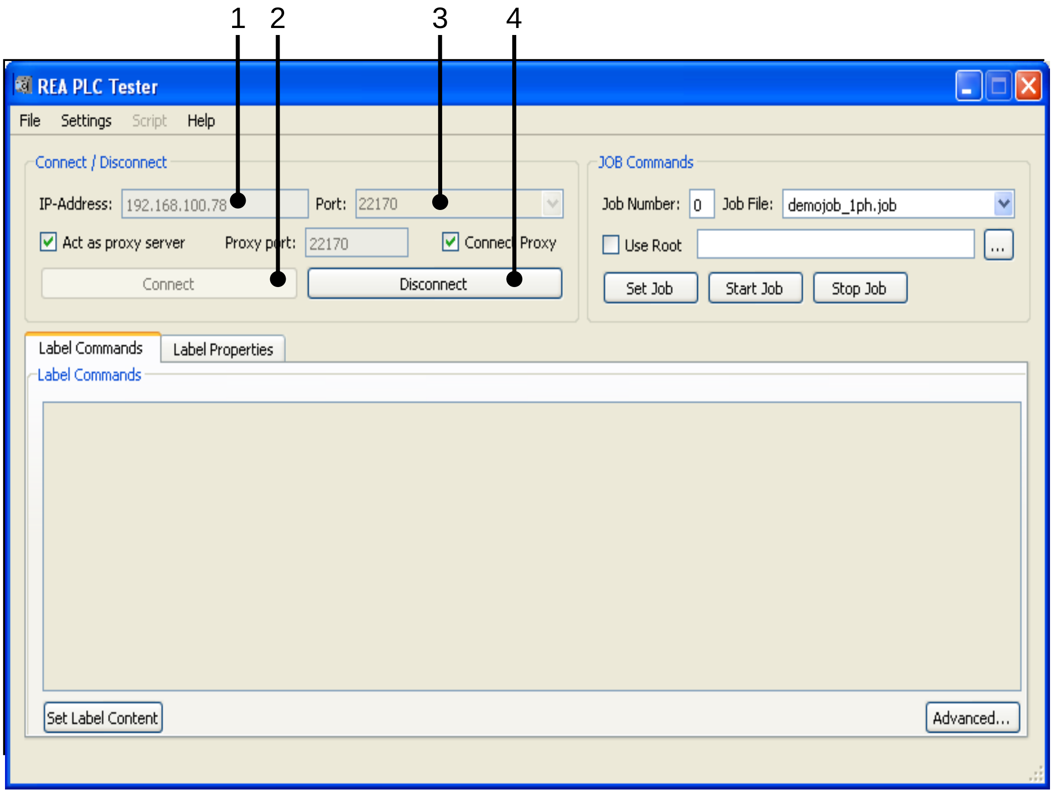
……

This program is intended for use in testing only: it is not designed for use in a production environment.

3.1.1 Connection setup (Connect)

After executing the program (.exe file), the main window appears – see Figure 1 on page 28. The information necessary to set up a connection can be entered here – see chap. 2.1 on page 9. The "Connect" button is used to attempt to establish a connection to the server. If the connection was successful, the "Connect" button is deactivated and the "Disconnect" button is activated instead. An error is returned after a timeout if no connection can be made.

If you wish to test communication with your client (PLC), one alternative is to establish a connection to the proxy server integrated into **REA-PLC Tester** – see chap. 0 on page 33.



IP-Address: 192.168.100.78

Port: 22170

☑ Act as proxy server

Proxy port: 22170

☑ Connect Proxy

Job Number: 0

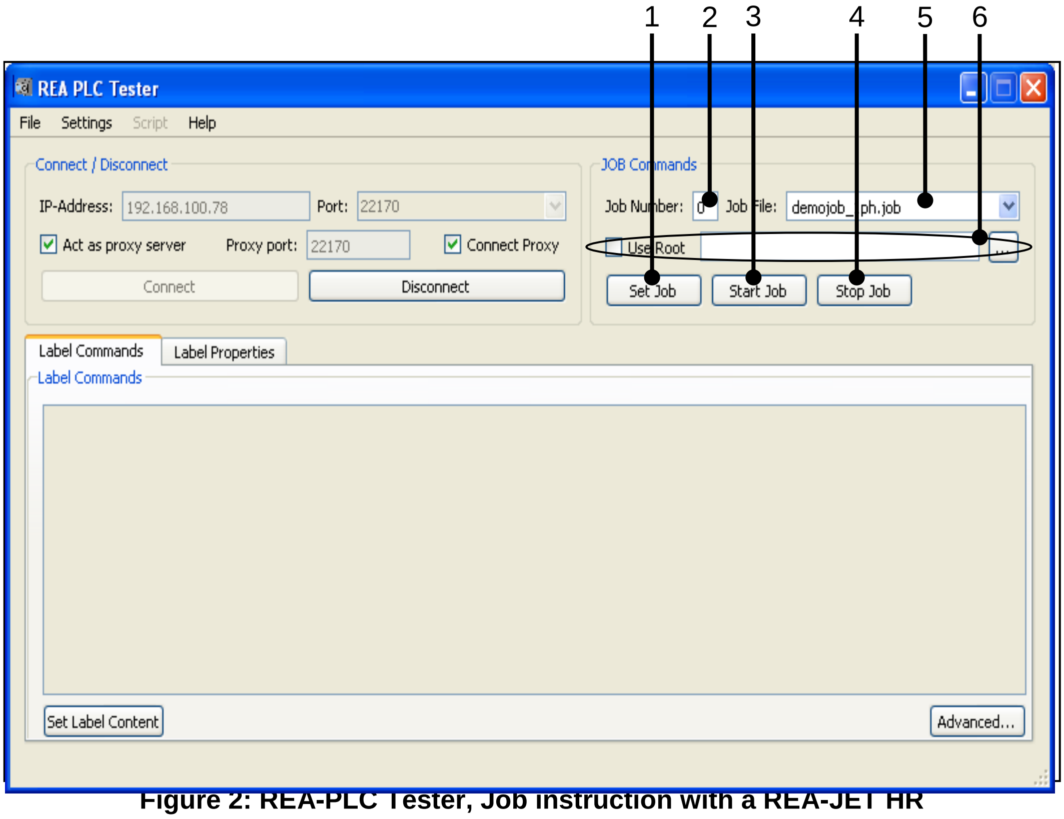
Job File: demojob_1ph.job

☐ Use Root

No.	Description
1	Server IP address (format: xxx.xxx.xxx.xxx)
2	Button to establish a connection with the server
3	Port number, see chap. 2.1.1 on page 10
4	Break connection to server

3.1.2 *Print job instructions (Set Job, Start Job, Stop Job)*

The icons in the "Job commands" windows are activated only once a connection has been established successfully. In Figure 2, we can see that a job file with the name "demojob_1ph.job" is available on the server.



IP-Address: 192.168.100.78 Port: 22170

☑ Act as proxy server Proxy port: 22170 ☑ Connect Proxy

Job Number: 0 Job File: demojob_1ph.job

☐ Use Root

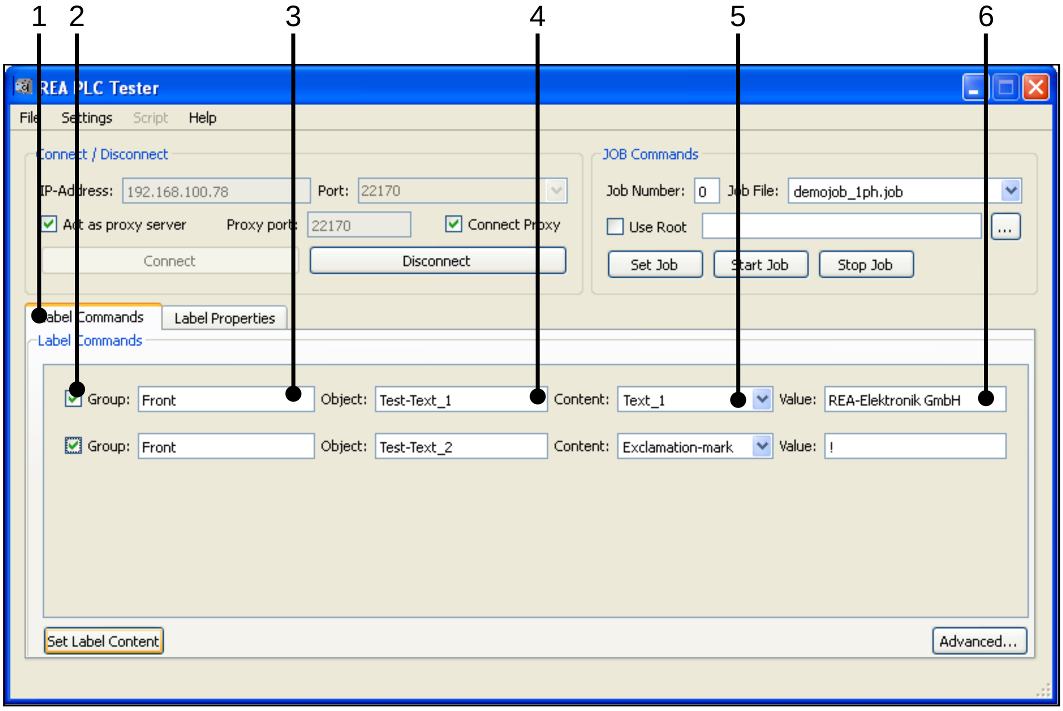
Figure 2: REA-PLC Tester, Job instruction with a REA-JET HR

No.	Description
1	Assign a print job, see chap. 2.3.1 on page 11
2	Assign to job no. 0 ≡ Job 1 Currently, only one job is supported
3	Release a text for printing, see chap. 2.3.2 on page 12
4	Cancel a print release, see chap. 2.3.3 on page 13
5	Select the print job (file name), see chap. 2.3.1 on page 11
6	Select print jobs from an external source (network or a local drive). See also chap. 0 on page 33

3.1.3 *Overwrite object contents (Set Label Content)*

Once the button "Set Job" has been selected, the job specified in the "Job File" field is then sent to the server, and editable content is then listed in the "Label Commands" window.

The individual fields can now be changed. Once edits are complete, the fields are then sent to the server and typeset via the button "Set Label Content". See also chap. 2.3.4 on page 13



Connect / Disconnect

IP-Address: 192.168.100.78 Port: 22170

☑ Act as proxy server Proxy port: 22170 ☑ Connect Proxy

JOB Commands

Job Number: 0 Job File: demojob_1ph.job

□ Use Root

Label Commands

☑ Group: Front Object: Test-Text_1 Content: Text_1 Value: REA-Elektronik GmbH

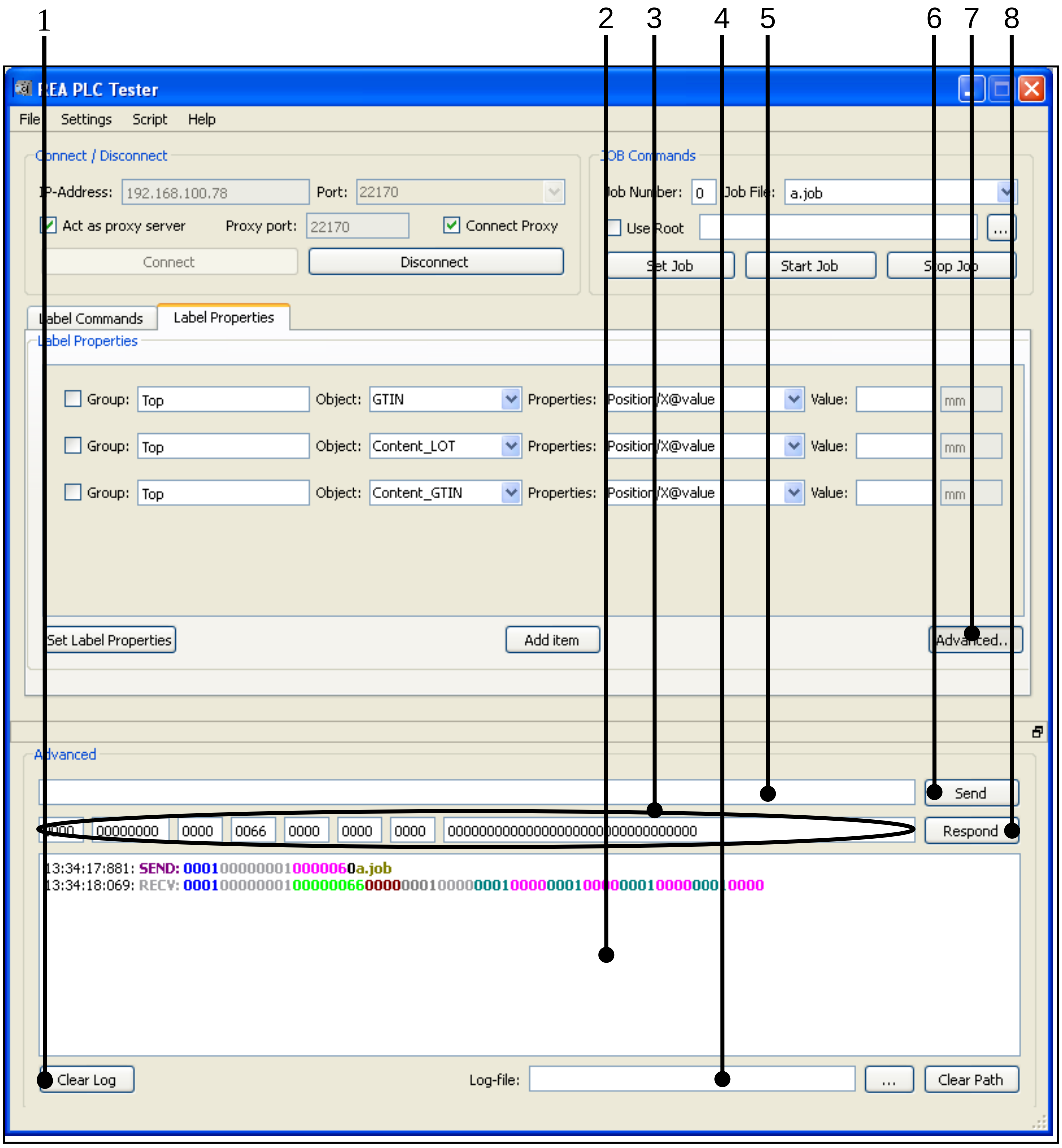
☑ Group: Front Object: Test-Text_2 Content: Exclamation-mark Value: !

Figure 3: REA-PLC Tester, Set Label Content with a REA JET HR

No.	Description
1	Only the content that was selected is then transmitted to the server with the "Set Label Content" button
2	"Set Label Content" button, for transmitting text content to the server
3	Name of the group
4	Name of the object
5	Name of the content
6	New content that is to be written

3.1.4 Advanced

This is an additional window, which opens when the "Advanced…" button is selected. This window can be used to follow the communication between the server and the client. Data strings can also be transmitted directly to the server. To do so, bytes are entered directly in the format `\0x04`, for example: `00020000000030000010\0x04` for `Start Job` with EOT on port 22169. The data string is sent to the server with `Send`.



IP-Address: 192.168.100.78 Port: 22170

☑ Act as proxy server Proxy port: 22170 ☑ Connect Proxy

Job Number: 0 Job File: a.job

☐ Use Root

Label Properties

☐ Group: Top Object: GTIN Properties: Position/X@value Value: mm

☐ Group: Top Object: Content_LOT Properties: Position/X@value Value: mm

☐ Group: Top Object: Content_GTIN Properties: Position/X@value Value: mm

0000 | 00000000 | 0000 | 0066 | 0000 | 0000 | 0000 | 00000000000000000000000000000000

```
13:34:17:881: SEND: 00010000000010000060a.job
13:34:18:069: RECV: 0001000000001000000660000000100000001000000010000000100000001000000010000
```

Log-file:

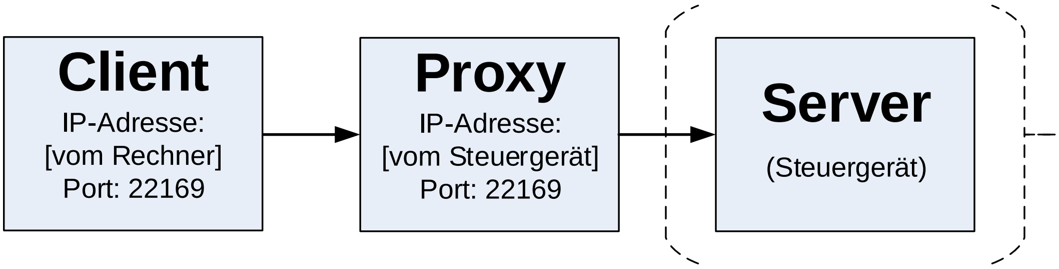
Figure 4: REA-PLC Tester, Advanced with a REA-JET HR

No.	Description
1	Erases the content of the log window – see no. 2
2	A log window, for use in following the communication between the server and the client
3	Input field for a response (64 bytes long), which can be sent using the "Respond" button
4	Specifies a file that will be used to store all of the entries displayed in the log window
5	Input field for an instruction that should be sent to the server – see also chap. 2.3 on page 11
6	This is used to send the data string that was entered in field 5
7	Opens the Advanced window
8	This is used to send the data string that was entered in field 3

3.1.5 *Proxy*

No control unit (server) is necessary for the purposes of familiarisation with the **REA-PLC** interface protocol or troubleshooting communication errors. Instead you can establish a connection over a proxy. In this case, you will handle server and client functions manually yourself. To find out how to do this, please consult chap. 2.19 on page 11 ff.

General setup of a proxy connection:



3.1.5.1 Sample configuration

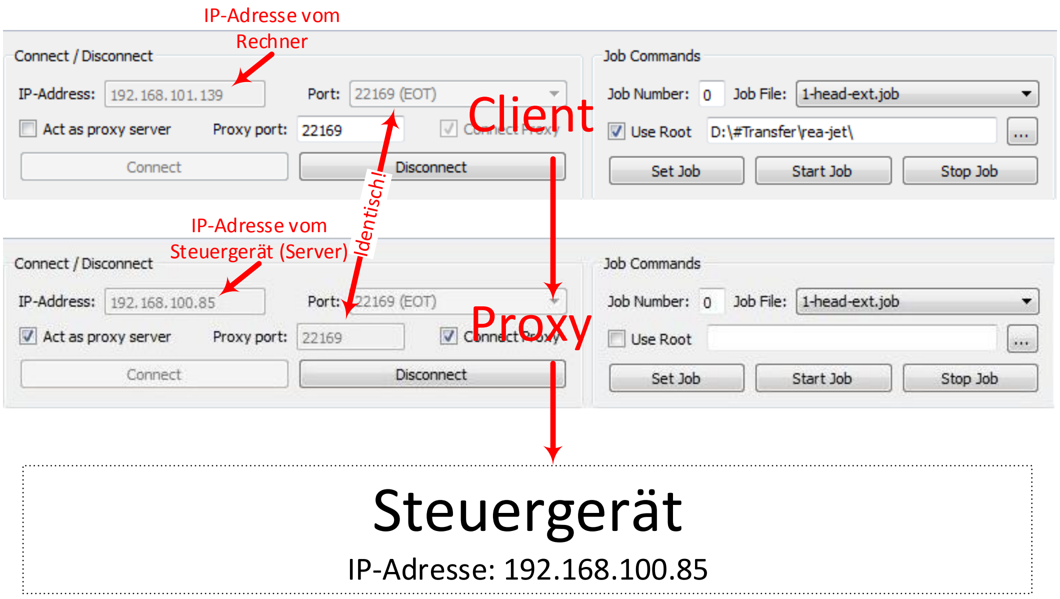
3.1.5.1.1 Sample client configuration:

- IP address: 192.168.101.139
  Computer's IP address
- Port [see chap.2.1 on page 9, or optional]

3.1.5.1.2 Sample proxy configuration:

- IP address: 192.168.100.78
  IP address of the server (control unit) – note that this is NOT necessary for a proxy-only connection!
- Port [see chap.2.1 on page 9]
- Act as proxy server: √
- Proxy port: [see "Client settings"]
- Connect proxy: [optional, recommended: √]

A real-world configuration is shown below:



Connect / Disconnect

IP-Address: 192.168.101.139 Port: 22169 (EOT)

□ Act as proxy server Proxy port: 22169 ☑ Connect Proxy

Job Commands

Job Number: 0 Job File: 1-head-ext.job

☑ Use Root D:\#Transfer\rea-jet\

Connect / Disconnect

IP-Address: 192.168.100.85 Port: 22169 (EOT)

☑ Act as proxy server Proxy port: 22169 ☑ Connect Proxy

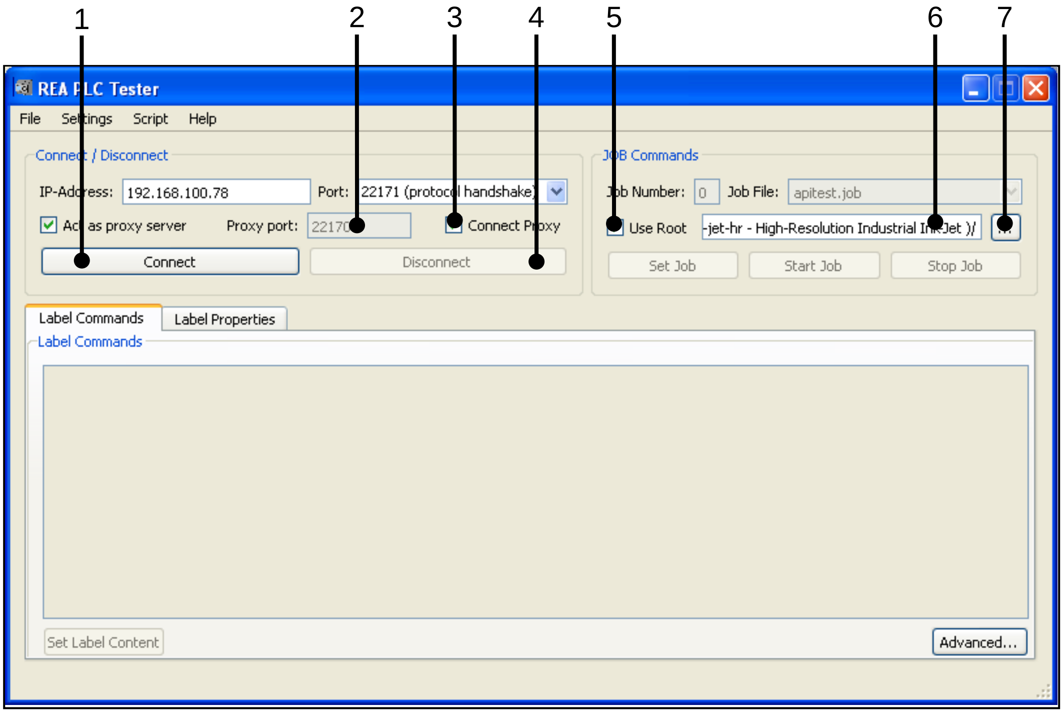
Job Commands

Job Number: 0 Job File: 1-head-ext.job

□ Use Root

3.1.5.2 Description of the proxy component GUI

The following items are required for a proxy configuration:



REA PLC Tester

Connect / Disconnect

IP-Address: 192.168.100.78 Port: 22171 (protocol handshake)

☑ Act as proxy server Proxy port: 22170 □ Connect Proxy

JOB Commands

Job Number: 0 Job File: apitest.job

□ Use Root -jet-hr - High-Resolution Industrial InkJet)/

Figure 5: REA-PLC Tester, Proxy connection with a REA JET HR

No.	Description
1	Set up a connection to the proxy server, on the client side Set up a connection to the server, on the proxy side
2	For a proxy connection, a port must be specified; this can be identical with the port on the server side
3	On the proxy side, a connection is set up or broken automatically, depending on the client – see no. 1 and no. 4
4	Break the connection (see no. 1)
5	Must be selected if the directory structure is being used from a server and only the root folder is being referenced (see no. 6)
6	Path for the data source folder where job files (and others) are stored
7	Open the window for selecting the data source folder

3.1.5.3 Step-by-step proxy connection guide

Please keep the following in mind for a proxy connection:

For this example, the "Samba" protocol was used to create a data copy from the server (control unit (**REA JET HR**)) to a local drive, so as to be able to assign a previously defined job.

3.1.5.3.1 Proxy configuration

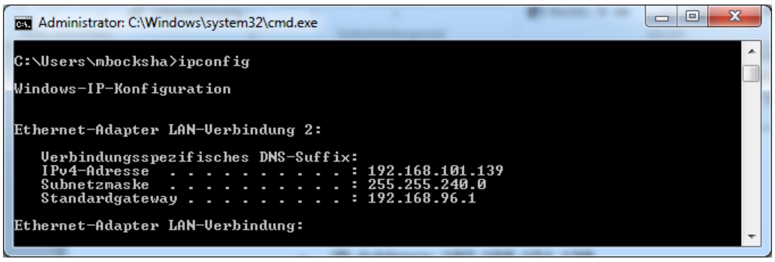
Start the REA PLC Tester application.

- Choose a directory – please see no. 7 in the figure no. 5 on page 36.
- Activate the "Use Root" check box (see no. 5, figure 5 on page 36). Assuming you have data in the directory, you should now be shown a job in the "Job File:" field. See the following figure, with the job file "apitest.job".
- A "Connect" is not necessary for a proxy-only connection. Accordingly, you do NOT need to enter an IP address from the server (control unit).

3.1.5.3.2 Client configuration

Start another REA PLC Tester (Client) instance.

- The first task is to discover the computer's IP address. One method for doing so (on Windows) is to launch the "cmd.exe" program on your PC and then issue the "ipconfig" command. This should give you the following information about your PC.



```
C:\Users\mbocksha>ipconfig

Windows-IP-Konfiguration


Ethernet-Adapter LAN-Verbindung 2:

   Verbindungsspezifisches DNS-Suffix:
   IPv4-Adresse  . . . . . . . . . . : 192.168.101.139
   Subnetzmaske  . . . . . . . . . . : 255.255.240.0
   Standardgateway . . . . . . . . . : 192.168.96.1

Ethernet-Adapter LAN-Verbindung:
```

In this specific example, the IP address is 192.168.101.139 (see "Sample configuration").

- Enter the IP address into the "IP Address" field.
- Select the same port as used for the proxy configuration.
- Now connect to the proxy by clicking the "Connect" button.

3.1.6 *Save settings*

To avoid having to enter the same settings more than once, you can save them by selecting `File` -> `Save settings` from the menu.

The settings saved include the values given for "IP Address", "Port", "Act as proxy server" and "Proxy port".

These values are stored in the Window Registry under `HKEY_USERS`, [], `Software`, `REA-Elektronik GmbH`, `REA-PLC Tester` using the current user account.

3.1.7 *Clear settings*

This command is used to erase/reset the settings configured as described in chap. 3.1.6 on page 35. The erase/reset does not take effect until the program is restarted.

4 titan-add-on-serial2plc

REA-Elektronik GmbH offers a software solution with which the data can be sent via the serial interface. Note that the data string must be terminated with 0x0D.

The answers from the server (response) will also be terminated with an additional 0x0D 0x0A.

Sample data sets for the "SerMoni.exe" terminal program, which can be supplied free-of-charge by **REA-Elektronik GmbH.**

```
“000400000000300001f00160;1;Test-Text_1;Text_10001A”
0D
```

```
“000400000000300001f00160;1;Test-Text_1;Text_10001B”
0D
```

If the control unit is ready to accept telegrams, the text message "REA-JET is READY 0x0D 0x0A" is output on the serial interface.

4.1 serial_parameter.ini

The file "serial_parameter.ini" is located in the [control unit's IP address]\rea-jet\service\" directory. This can be used for the subsequent modification of the serial interface parameters. The control unit must be restarted after making changes to the parameters.

The first serial interface can be named as follows, depending on the device type:

- /dev/ttyS0
- /usr/reajet/dev/ttyS0

The serial interface can be configured as shown in the sub-sections below:

4.1.1 9600/8/N/1

stty -F /usr/reajet/dev/ttyS0 9600 intr 00 quit 00 erase 00 kill 00 ixany -parenb -parodd cread -icanon min 0 time 0

4.1.2 19200/8/E/1

stty -F /usr/reajet/dev/ttyS0 19200 intr 00 quit 00 erase 00 kill 00 ixany parenb -parodd cread -icanon min 0 time 0

4.1.3 115200/8/O/1

```
stty -F /usr/reajet/dev/ttyS0 115200 intr 00 quit 00 erase 00 kill 00
ixany parenb parodd cread -icanon min 0 time 0
```

5 Notes