TEC-IT

Datenverarbeitung GmbH

Wagnerstr. 6
A-4400 Steyr

TBarCode
FOR
Unix / Linux

Version 1.2-1

**Linux, FreeBSD,
AIX 4.x/ 5.x,
HP-UX 11.00,
and others**

Documentation

Last Update 16-Feb-2005

e: office@tec-it.com
w: www.tec-it.com

p: +43 / (0) 7252 / 72 720
f: +43 / (0) 7252 / 72 720 - 77

1.1 Table of Contents

Table of Contents

Table of Contents

2 About TBarCode for UNIX …

2.1 Features

TBarCode for UNIX

- reduces the costs for bar code printing, because the licensing scheme does not depend on the number of printers.
- makes it possible to print bar codes on **any PCL or PostScript printer**, without the need of installing a bar code extension cartridge or a special bar code font. Thus you can print bar codes in a complete device independent way.
- works in a completely transparent way. Bar codes are computed by a daemon process in the background.
- is available as precompiled bar code-printing engine for LINUX, AIX and HPUX.
  Other operating systems on request.

2D Symbologies

Beside linear bar codes (e. g. 2of5, 2of5 ITL, Code39, Code128, EAN128, EAN, UPC...) **TBarCode for UNIX** supports also 2D symbologies like:

- PDF417
- Data Matrix
- MaxiCode

These 2D-symbologies features very high data capacities with enhanced data security and are required by several enterprises for their documents (and labels) – a selection:

- MaxiCode by UPS®
- PDF417 by General Motors®
- PDF417 and MaxiCode by the AIGA (B-10, Automotive Industry Action Group).

Bar Code Quality

TBarCode for UNIX offers the possibility to specify **all** bar code parameters – these are for example:

- Specification of the module width in absolute units (device independent)
- PDF417 format and properties like error correction level
- Selection of the subsets of Code128 (subsets A, B and C – and automatic compression mode)
- The bar codes are created in vector graphic format (EPS and PCL), which uses the maximum of the available printing resolution.
- And many others…

2.2 System Requirements

Platforms:	LINUX (tested with SuSE and RedHat)	CPU:	Intel (x86)
Platforms:	AIX 4.3	CPU:	PowerPC
Platforms:	HP-UX 11.00	CPU:	PA-RISC 2.0 (others on request)
Platforms:	LINUX (tested with SuSE and RedHat)		

Output Devices	Postscript Level 2		
Output Devices	PCL Level 5		

2.3 Contact Information

TEC-IT Datenverarbeitung GmbH
Wagnerstr. 6
A-4400 Steyr (Austria)

Phone: +43 / 7252 / 72720

Fax: +43 / 7252 / 7272077

www.tec-it.com

mailto:barcode@tec-it.com

3 Installation TBarCode for Unix

3.1 Installing the binaries

Basic Workflow

The necessary steps to print bar codes are described in the following sections (use it in the correct order).

Table 1: Basic Workflow for Printing Bar Codes

Step	Operation	Optional	OS commands or subsystems
1	Setup TBarCode for UNIX	No	Rpm or kpackage or gnorpm
2	Testing TBarCode Client	Yes	tbarcodeclient
3	Installing TBarCode Filter	No	/etc/printcap
4	Testing TBarCode Filter	Yes	lpr, tbarcodefilter
5	Implementing Control Sequences	No	depends on your application
6	License TBarCode	No	license.ini

How to install the Linux Version

To setup **TBarCode for UNIX** install the RPM binary packages in the following order:

For the installation you may use the rpm command or any package manager that comes with your LINUX distribution, e.g. gnorpm, kpackage.

- ➔ **libtbarcode-1.1.0-1.i386.rpm**
  This package contains the shared library, which is necessary to create bar codes. The unlicensed version of this product draws an additional horizontal bar over the generated code. Usually the bar code will still be readable by scanners.
- ➔ **tbarcoded-1.2-1.i386.rpm**
  This package contains the **TBarCode Daemon**, which handles the creation of the bar codes. The daemon is required for **TBarCode Client** and **TBarCode Filter**.
- ➔ **tbarcodeclient-1.2-1.i386.rpm**
  This package contains the **TBarCode Client** and an ini-file for the client (barcode.ini).
- ➔ **tbarcodefilter-1.2-1.i386.rpm**
  This package contains the **TBarCode Filter** and additional scripts.
- ➔ Run '**ldconfig /usr/local/lib**'.
- ➔ The setup of **TBarCode for UNIX** is completed.

SUID-Bit: After setup of TBarCode change to the installation path (/usr/local/tbarcode) and make sure that the executables have the User-ID (SUID Bit) set. See next chapter for details.

How to install the AIX and HP-UX Version

The downloaded package file contains the binaries and supplementary files of **TBarCode for AIX (HP-UX)**. Unzip the package and extract the files to the directory `/usr/local/tbarcode`.

After this step **TBarCode Client** and **TBarCoder Filter** can be used by calling:

`/usr/local/tbarcode/tbarcodeclient` *Parameters* or

`/usr/local/tbarcode/tbarcodefilter` *Parameters*

Execute Rights + SUID Bit

Make sure that the following files have execute rights and the User-ID (SUID) bit is set.

```
-rwsr-xr-x   1 root       users       413696 Oct 13 17:04 tbarcodeclient
-rwsr-xr-x   1 root       users       962560 Oct 13 17:04 tbarcoded
-rwsr-xr-x   1 root       users       425984 Oct 13 17:04 tbarcodefilter
```

Missing attributes may be set (by root only) with: `chmod a+xs name_of_file`

Execute Rights

The following files need execute rights:

```
-rwxrwxrwx   1 root       users         243 Oct 13 17:03 stoptbarcoded
-rwxr-xr-x   1 root       users        1054 Oct 13 15:40 tbarcodescript
-rwxr-xr-x   1 root       users        1468 Oct 13 15:40 tbarcodescript_fwd
```

Missing attributes may be set with: `chmod a+x name_of_file`

→ If you have problems, please contact our free support mailto:support@tec-it.com

How to install the Debian Version

Assuming you have downloaded the TBarCode 1.2 setup files for Debian

Type the following commands in your shell:

```
tar xzf TBarCode-1.2-debian3.0r2.tar.gz
cd TBarCode-1.2
./install.sh
```

Installation will look like this (in your shell)

```
debian:~# tar xzf TBarCode-1.2-debian3.0r2.tar.gz
debian:~# ls -l
total 408

drwxr-sr-x    5 root     staff     4096 Jun 15 14:19 TBarCode-1.2
-rw-r--r--    1 root     root    406197 Jun 15 14:58 TBarCode-1.2-debian3.0r2.tar.gz

debian:~# cd TBarCode-1.2
debian:~/TBarCode-1.2# ls -l
total 20

drwxr-sr-x    2 root     staff        4096 Jun 15 11:26 include
-rwxr--r--    1 root     staff        2363 Jun 15 14:31 install.sh
drwxr-sr-x    2 root     staff        4096 Jun 15 14:25 lib
drwxr-sr-x    2 root     staff        4096 Jun 15 11:30 tbarcode
-rwxr--r--    1 root     staff         856 Jun 15 14:36 uninstall.sh

debian:~/TBarCode-1.2# ./install.sh

TBarCode for Linux - Installation

Copying include files...
Copying libraries...
Copying tbarcode files...
```

```
Registering TBarCode Library...
Creating link for TBarCode Client...
Setting file permissions...
Installation finished.

debian:~/TBarCode-1.2#
```

→ After installation, make sure that the SUID-Bit is set correctly (see previous chapters for details).

Uninstallation

Type the following commands in your shell:

```
tar xzf TBarCode-1.2-debian3.0r2.tar.gz
cd TBarCode-1.2
./uninstall.sh
```

→ If you have problems, please contact our support mailto:support@tec-it.com

3.2 Setup TBarCode Filter

This section covers the “standard” spool systems – for configuring CUPS see next section.

Printer Queue Setup - Linux (LPRng)

If printed files should be automatically filtered with **TBarcode Filter**, some changes have to be made to /etc/printcap:

1. Search through /etc/printcap for the entry of the printer queue you want to use with **TBarCode Filter**. This entry may look similar to this example:

```
printer:\
    :sh:\
    :ml=0:\
    :mx=0:\
    :sd=/var/spool/lpd/printer:\
    :af=/var/spool/lpd/printer/printer.acct:\
    :lp=/dev/lp0:\
    :lpd_bounce=true:\
    :if=/usr/share/printconf/util/mf_wrapper:
```

   If the printcap entry doesn't contain the parameter if=... then perform step 2 otherwise go to step 3.

2. Add the following to the printcap entry:

```
…printcap entry… \
    :lpd_bounce=true:\
    :if=/usr/local/tbarcode/tbarcodescript:
```

   Go to step 4.

3. Remember the original printer filter (in this example `/usr/share/printconf/util/mf_wrapper`) and change the entry to:

```
printer:\
    :sh:\
    :ml=0:\
    :mx=0:\
    :sd=/var/spool/lpd/printer:\
    :af=/var/spool/lpd/printer/printer.acct:\
    :lp=/dev/lp0:\
    :lpd_bounce=true:\
    :if=/usr/local/tbarcode/tbarcodescript_fwd:
```

   In `/usr/local/tbarcode/tbarcodescript_fwd` substitute "path_of_original_filter" with the name of the original printer filter (in this example `/usr/share/printconf/util/mf_wrapper`).
   Go to step 4.

4. Restart lpd: Call `/etc/rc.d/init.d/lpd restart`

Normally **TBarCode Daemon** will be started automatically if it is not running. In some cases the printer queue hangs if **TBarCode Daemon** is not running. In these cases you have to start the daemon before printing by calling **TBarCode Filter** *with the same parameters as in the printer queue*, e.g.

```
/usr/local/tbarcode/tbarcodefilter </dev/null >/dev/null
```

You may want to include this call in your `/etc/rc.d/rc.local` file to start the daemon automatically when the system boots.

Printer Queue Setup - AIX

To install **TBarCode Filter** in the printer queue follow these steps:

1. Choose in which printer queue you want to use **TBarCode Filter**.
2. Assign **TBarCode Filter** as a user-defined filter in the virtual printer definition of this queue:
   - Use the tool "`lsvirprt`" to edit the attributes of the virtual printer definition. Attributes f1, f2, f3, f4, f5 may specify user-defined filters.
   - Set the value of f1 to "`/usr/local/tbarcode/tbarcodefilter Parameters`" , e.g. `f1=/usr/local/tbarcode/tbarcodefilter`

If you want to print a file and filter bar codes, call `qprt` with the paramter `-f1`, e.g.

```
qprt -PQueuename -f1 /usr/local/tbarcode/testfile.ps
```

Normally **TBarCode Daemon** will be started automatically if it is not running. In some cases the printer queue hangs if **TBarCode Daemon** is not running. In these cases set the value of f1 to "`/usr/local/tbarcode/tbarcodefilter Parameters </dev/null 2>/dev/null; /usr/local/tbarcode/tbarcodefilter Parameters`".

To permanently select **TBarCode Filter**, use `lsvirprt` to edit the virtual printer definition to set the value of the attribute _f to "1". Thus all print jobs for this queue will be filtered with **TBarCode Filter.**

TBarCode Filter with Unispool® (Holland House B.V.)

Create a script `/home/unispool/tbc_filter_script` with following content (the parameter –S is only necessary for Postscript-output from SAP R/3®):

```
tbarcodefilter -S </dev/null 2>/dev/null
cat $6 | /home/unispool/tbarcodefilter -S | /home/unispool/cexpand
```

In this case `/home/unispool` contains the tbarcodefilter binary or a link to the binary.

In Unispool® use the filter by calling `/home/unispool/tbc_filter_script`.

Printer Interface Script Setup - HP-UX

Here we describe how to setup the filter in the **standard lp spooler** coming with HP-UX 11. Please read the chapter Unix Printing (HP-UX and Solaris) in the appendix for background information.

First you should perform a basic test to see if the filter works on your system.

Enter the following command in your shell:

```
/usr/local/tbarcode/tbarcodefilter </usr/local/tbarcode/testfile.ps >result.ps
```

→ a result.ps file should have been created

HP-UX 11.xx can use the **lp** spooler or the **HPDPS** spooler (both adjustable with SAM). If you use LPRng (also available for HP-UX) the installation procedure would be the same as for Linux. HPDPS is also supported by TBarCode Filter -> please contact our support if you need help.

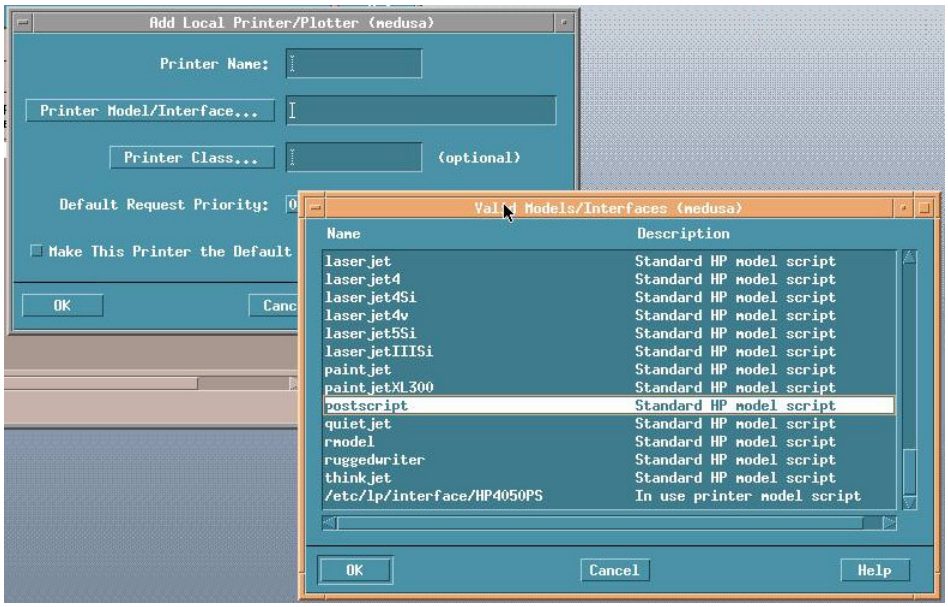
Below we focus on the SVR4 based lp spooler.

Using a local printer

TBarCode Filter can be integrated to the "model files" located in /usr/lib/lp. These files are scripts, which handle and describe the characteristics supported by the printer. Either you can add an own model file to this directory or modify an existing one.

It is very easy to call the filter inside such a script: each time a printout is made the filter will be called (because the script is run for each spool / job file).

Which model file/script is used (and has to be modified) depends on the settings within SAM: it is adjusted in the input field "printer model / interface".



You need to add some commands to the model script in order to call TBarCode Filter – see next headline, they are the same as with remote printers.

Using a remote printer

For a remote printer (LPR) there is another script responsible for calling the filter.

For instance, if you have a remote printer HP4050PS - the script, which handles the print-out is located in /etc/lp/interface - please check if you have a script in this place.

Then edit the script and insert the following lines before the final rlp command:

```
############################################################
# start TBarCode filter
#
# create temp file name
INPUT="$(mktemp /tmp/tbarcodefilter.$$.XXXXXX)"
# call tbarcodefilter to insert barcode, output to temp file
/usr/local/tbarcode/tbarcodefilter <$1  >$INPUT
# move temp file to original spool file (overwrite it)
mv $INPUT $1
# end filter
############################################################
/usr/sbin/rlp -I$requestid $BSDC $BSDJ $BSDT $BSDi $BSD1 $BSD2 $BSD3 $BSD4 $BSDw ...
```

Result: the filter is called before the spool job is sent to the printer with the rlp command

Make a test call with: `lp -d Printer /usr/local/tbarcode/testfile.ps`

→ on a Postscript printer you should see bar codes!!

3.3 Setup CUPS Filter

Some background info: CUPS is based on IPP (Internet Printing Protocol) and supports also "Remote-Printing". So with CUPS you can build up a Print Server.

The tools for printing with CUPS are called like LPRng, like as “lpr” or “lp”.

For example: lpr -P queue@server document

CUPS can understand the old LPD protocol; if the remote server is using the old printing system “LPRng”, it can nevertheless print on a CUPS-queue on a foreign printer (same input like above).

Important when using TBarCode: “raw queues” ignore all filters, so don’t use a raw queue for TBarCode. You have to use a printer driver (*.ppd).

CUPS also support the JetDirect protocol. IPP, LPD and JetDirect are realised with so-called „back ends", which are at the end of the filter queue and are responsible for sending the print job to the printer.

Setup CUPS for Postscript bar code filter

To install **TBarcode Filter** for CUPS some changes have to be made to `/etc/cups/mime.convs`:

1. Search through `/etc/cups/mime.convs` and look for the following entry:
   application/postscript applications/vnd.cups-postscript 66 pstops
   Replace “`pstops`” with “`/usr/local/tbarcode/tbarcodescript_cups`”, such that the line looks like this:
   application/postscript applications/vnd.cups-postscript 66 /usr/local/tbarcode/tbarcodescript_cups

2. Restart the printing service: Call `/etc/rc.d/init.d/cups restart`
3. To test your settings you can print a test-page:
   `lp -d` *`name_of_printerqueue`* `/usr/local/tbarcode/testfile.ps`

   The printout should contain some barcodes.

After these changes all *Postscript* files will be filtered. Bar codes are inserted it the Postscript data contains the TBarCode control sequences in the correct way. Please send us the spool files for analyzing if you can’t get your data filtered (Email: support@tec-it.com).

Setup CUPS for PCL bar code filter

In this description we want to build up a filter for PCL printing from SAP®. So our mime type is named “sap.pcl”. This method works for all PCL output, not only from SAP®.

Step 1

Add a new mime type application/sap.pcl in the file etc/cups/mime.types

```
...
application/msword        doc string(0,<D0CF11E0A1B11AE1>)
application/pdf           pdf string(0,%PDF)
application/postscript    ai eps ps string(0,%!) string(0,<04>%!)
application/vnd.hp-HPGL   hpgl string(0,<1B>&)\
                          string(0,<1B>E<1B>%0B) \
                          string(0,<1B>%-1B) string(0,<201B>)\
                          string(0,BP;) string(0,IN;) string(0,DF;) \
                          string(0,BPINPS;) \
                          (contains(0,128,<1B>%-12345X) + \
                           (contains(0,1024,"LANGUAGE=HPGL") \
```



```
                              contains(0,1024,"LANGUAGE = HPGL")))
application/sap.pcl          (string(0,<1B>E) + !string(2,<1B>%0B)) \
                              string(0,<1B>@) \
                              (contains(0,128,<1B>%-12345X) + \
                               (contains(0,1024,"LANGUAGE=PCL") \
                                contains(0,1024,"LANGUAGE = PCL")))

...

application/octet-stream
```

Not to be changed: the type application/vnd.cups-raw must remain in the file and also the application/octet-stream. The new type “application/sap.pcl” has more priority and will be handled first.

Step 2

Add a new filter for application/sap.pcl to the file mime.convs at third position:

```
application/pdf            application/postscript          33  pdftops
application/postscript     application/vnd.cups-postscript 66
                           /usr/local/tbarcode/tbarcodescript_cups
application/sap.pcl        application/vnd.cups-raw        66
                           /usr/local/tbarcode/tbarcodescript_pcl
…
…
application/octet-stream   application/vnd.cups-raw        0   -
```

At last position insert application/octet-stream to support direct raw printing (only if needed).

Important:

This works only if the printer is configured with a printer driver. That means that a CUPS driver file printername.ppd (e.g. deskjet.ppd) must be used in the CUPS setup.

If the printer is configured as raw printer or “raw print queue” (without a driver) it doesn’t work!!

Step 3:

Make backup copies of all these files, because if CUPS is reinstalled you need to reconfigure the files.

Step 4:

The script tbarcodescript_pcl inserts the bar code commands to the PCL data stream.

Here is the script **tbarcodescript_pcl** which must be copied to /usr/local/tbarcode and made executable.

```
#! /bin/bash

#_____________________________________________________________________________#
#                                                                             #
#                          Copyright (C) 2003 by                              #
#                    -- TEC-IT Datenverarbeitung GmbH --                      #
#   -- team for engineering and consulting in information technologies --     #
#                                                                             #
#                          All rights reserved.                               #
#_____________________________________________________________________________#
```



```
#                                                                              #

#______________________________________________________________________________
#
# Filter printjob file with tbarcodefilter and then filter it with original
# cups filter (pstops).
#______________________________________________________________________________
#
# if $6 is empty then printjob is read from stdin
# otherwise parameter $6 contains filename of printjob
#

exitvalue=

if test -z $6
then
  # no filename specified
  # => read from stdin

  /usr/local/tbarcode/tbarcodefilter <&0 >&1
  exitvalue=$?

else
  # filename specified
  # => read from file

  # make a temporary file for the output of tbarcodefilter
  FILTEREDFILE="$(mktemp /tmp/tbarcodefilter.$$.XXXXXX)"

  /usr/local/tbarcode/tbarcodefilter <$6 >$FILTEREDFILE
  cat $FILTEREDFILE
  exitvalue=$?

  # remove FILTEREDFILE file
  rm $FILTEREDFILE

fi

exit $exitvalue
```

Step 5:

Now, if PCL data is printed with TBarCode specific control sequences ($_tbcs…), the filter should be called (tbarcodescript_pcl) and bar codes are inserted to the printer data.

4 Printing Bar Codes

4.1 Available Tools

TBarCode for UNIX consists of a set of tools to create bar codes:

- **TBarCode Library** is a shared library that contains the functions to create bar codes. The library is used only on Linux platforms.
- **TBarCode Client** is a command line program that allows you to create bar codes and to save them in EPS or PCL files.
- **TBarCode Filter** reads a file from standard input (stdin) and writes the filtered result to standard output (stdout). It searches for bar code control sequences and replaces them with bar codes.
- **TBarCode Daemon** is a daemon program. It handles the creation of bar codes. **TBarCode Daemon** is used by **TBarCode Client**/**Filter** and will be started automatically if it is needed.

4.2 Using TBarCode Client

Test of TBarcode Client

TBarCode Client is a command line program that allows you to create bar codes and to save them in EPS or PCL format.

To test **TBarCode Client**, do the following:

1. Call `tbarcodeclient -fbarcode.eps b20 dTest123`
2. View the generated bar code `barcode.eps` with any PS viewer (e.g. gv, ggv, kghostview) If **TBarCode for UNIX** is not licensed, a black horizontal bar will be drawn across the bar code. Normally the created bar code will still be readable by most scanners.

Arguments of TBarCode Client

The executable is named tbarcodeclient and is called with following arguments:

```
tbarcodeclient Options BarcodeSettings
```

Options:

Parameter	Interpretation
-h	Help information and available command line argruments.
-f*Filename*	Write output to specified file. If option -f is missing, output will be written to standard output (stdout).
…	For more parameters please see Barcode Control Sequences

Bar code Settings:

Determine which bar code shall be printed. Bar code settings are specified with the same parameters used for the bar code control sequences (but without $_tbcs/$_tbce). See Barcode Control Sequences / Command Line Parameters for detailed information about available command line parameters.

Examples:

- Show command line arguments information:
  `tbarcodeclient -h`
- Create a bar code (Bar code type: Code128, content: “This is a test bar code”) and write postscript output to stdout:
  `tbarcodeclient b20 d”This is a test bar code”`
- Create a bar code (Bar code type: PDF417, content “123abc”, width: 40 mm, height: 20 mm, output: PCL) and store it in `barcode.pcl`:
  `tbarcodeclient -fbarcode.pcl tPCL b55 w40 h20 d123abc`

LibTBarCode – Dependencies

Listed for Redhat 9 (Kernel version 2.4.20.)

Libraries:

```
libc.so.6 (libc-2.3.2.so)
libm.so.6 (libm-2.3.2.so)
libstdc++.so.5 (libstdc++.so.5.0.3)
ld-linux.so.2 (ld-2.3.2.so)
```

4.3 Using TBarCode Filter

Test of TBarCode Filter

To test if **TBarCode Filter** is working, perform these steps:

1. Call `/usr/local/tbarcode/tbarcodefilter </usr/local/tbarcode/testfile.ps >result.ps`
2. View `result.ps` in any postscript viewer. A set of bar codes should appear in this file.

To test if **TBarCode Filter** was installed properly in printcap, you may print a test sheet:

1. Call `lpr -P` *`printerqueue`* `/usr/local/tbarcode/filtertest.ps`
   (where printerqueue is the name of the queue, where TBarCodeFilter was installed)
2. Check if the printout contains the bar codes.

Arguments of TBarCode Filter

The executable is named `tbarcodefilter`.

All parameters are optional. Normally `tbarcodefilter` is called without parameters.

For some parameters the **TBarCode Daemon** has to be restarted before they take effect. To stop **TBarCode Daemon** call: `/usr/local/tbarcode/stoptbarcoded`
The Daemon will be started automaticallly by **TBarCode Filter**.

Options:

Parameter	Interpretation	Restart Daemon?
-h	Help information and available command line arguments	no
-c	Create reset command for PCL files. (PCL reset command will be printed at the beginning and at the end of the file).	no
-d*PathOfDaemon*	Full path of tbarcoded	no
-f*Errorfile*	Full path of error file	yes
-i*ID*	Unique ID for inter-process communication.	yes
-l	Log error messages using the syslog facility (syslogd)	yes
-m*Size*	Size of shared memory in bytes	yes
-p*Path*	Path of the directory of TBarCode for UNIX	yes
-r*Retries*	Number of retries to connect to TBarCode Daemon	no
-R	Remove barcode escape sequence. (Default is to override the barcode escape sequence with blanks.)	no
-s	Do not write error messages to stderr	yes
-S	SAP-Flag: Remove surrounding parantheses of bar	yes

	code data in SAP Postscript documents.	
-t*Timeout*	Timeout in seconds	no
-x	Print debug messages	yes

5 Barcode Parameters

5.1 Control Sequences for Filter and Client

TBarCode Filter reads a file and searches for special control or control sequences (similar to SAP R/3 Printcontrols). If such a sequence is found, the sequence will be replaced with a bar code.

TBarCode Client uses the same parameters for the command line arguments.

Examples for bar code control sequences:

$_tbcs a n g0 b16 w300 h100 c0 d"Test123"$_tbce

$_tbcs b20 d123456$_tbce

$_tbcs tPS b20 d123456$_tbce

$_tbcs tPCL d123456$_tbce

$_tbcs tPS b55 w50 h30 Pe4 d"This is a PDF417"$_tbce
$_tbcs tPS b20 R1:2:3:4:1:2:3:4 d123456$_tbce

The file `/usr/local/tbarcode/testfile.ps` is an example for a postscript file which contains a bar code control sequence.

Standard Parameters

Parameter	Description
$_tbcs	Marks the beginning of the sequence (used with filter)
$_tbce	Marks the end of the sequence (used with filter)
d*Content*	Content = data of bar code; **must be the last parameter** before $_tbce
x*Position*	Absolute x position in mm (* see above)
y*Position*	Absolute y position in mm (* see above)
w*Width*	Width of bar code in mm (e.g. w50 or w53.12)
h*Height*	Height of bar code in mm
ot	Orientation: Top (x/y-Position sets the upper left corner of the barcode. Default in Postscript.)
ob	Orientation: Bottom (x/y-Position sets the lower left corner of the barcode. Default in PCL.)

b*BarcodeNo*	Number of bar code (see Bar code types in the Appendix)
c*MethodNo*	Number of checkdigit calculation method (see Check digit calculation method in the Appendix)
r*Rotation*	Rotation in degrees (0, 90, 180 or 270)
T(on|off)	Show text.
a	Print text above bar code (default is below)
s*(*on*/*off*)*	Translate escape sequences in input data (see Encoding bytes and control characters in input data in the Appendix).
A*(*on*/*off*)*	Turn auto correct on or off
g*GuardWidth*	Width of guarding line in mm
f*Fontname*	Fontname in Postscript or Typeface Family Value in PCL, e.g. Postscript: Times-Roman, Courier, Helvetica, … PCL: 4101, 4099, 16602, … If the number from the f parameter is 1000 or bigger than 1000, it will be identified as PCL-Font number. PCL Font table: PCL Font Numbers
f*Fontsize*	Size of font in points.
t*Format*	Output format: PS (=Postscript, default) or PCL
i*Distance*	Text distance in mm
N*Height*	Notch height in mm
m*ModWidth*	Module width (narrow bar width) in µm (= 1/1000 mm), if used the W parameter for the symbol width is irrelevant.
R*Ratio*	Print ratio
F*Format*	Format string used for formatting bar code data prior to printing it. (see Formatting bar code data in the Appendix)
O	Calculate optimal width of bar code
Qh*HorzQZ*	Horizontal quiet zone in mm (e.g. Qh1.34 or Qh5). The specified quiet zone is a blank space, which is added to the left and right side of the symbol. Usually the Quiet zone should be 10 times the module width or higher.
Qv*VertQZ*	Vertical quiet zone in mm (e.g. Qv1.34 or Qv5). The specified quiet zone is a blank space, which is added to the top and bottom of the symbol. Usually the Quiet zone should be 10 times the module width or higher.
I	Use *initgraphics* command in postscript. This may improve the positioning of the bar code if relative positioning is used in Postscript documents.
e	Move cursor to end of barcode in PCL.

W	Remove leading and trailing spaces from content.

Control sequence for PDF417 bar codes:

Parameter	Description
Pr*Number*	Number of rows
Pc*Number*	Number of columns
PR*Ratio*	Row-columns ratio
PRauto	Automatically choose row-column ratio.
Ph*Height*	Height of row in mm
Pe*Number*	Number of error correction level

Control sequence for Macro PDF417 bar codes:

Parameter	Description
Pi*Index*	Segment index
Pf*FileId*	File id
Pl	Last segment
Pn*Filename*	File name
Px*Segcount*	Segment count
Pt*Timestamp*	Time stamp
Ps*Sender*	Sender
PaAddressee	Addressee
Pz*Filesize*	File size
Pm*Checksum*	Checksum

Control sequence for Data Matrix bar codes:

Parameter	Description
Ds*Size*	Size (*Size* is taken from the table Data Matrix symbol sizes)
Df*Format*	Format (*Format* is taken from the table Data Matrix Format)
Dr	Draw rectangular instead of square
Das*Sum*	Structured Append: Sum
Dai*Index*	Structured Append: Index

Daf*File*	Structured Append: File ID

Control sequence for MaxiCode bar codes:

Parameter	Description
Mm*Mode*	Mode (*Mode* is taken from the table MaxiCode Mode)
Mu*Undercut*	Undercut of hexagon (0-100; default is 75)
Mp*Preamble*	Preamble
Mas*Sum*	Structured Append: Sum
Mai*Index*	Structured Append: Index
Mss*Serviceclass*	Structured Carrier Message (SCM): Service Class
Msc*Countrycode*	Structured Carrier Message (SCM): Country Code
Msp*Postalcode*	Structured Carrier Message (SCM): Postal Code

Control sequence for QRCode bar codes:

Parameter	Description
QRv*Version*	Version (Symbol Size) (*Format* is taken from the table QRCode Version)
QRf*Format*	Format (*Format* is taken from the table QRCode Format)
QRi*Indicator*	Format Application Indicator
QRe*ECLevel*	Number of Error Correction Level (*ECLevel* is taken from the table QRCode ECLevel)
QRm*Mask*	Mask Pattern (0-7)
QRas*Sum*	Structured Append: Sum
QRai*Index*	Structured Append: Index
QRap*Parity*	Structured Append: Parity Byte

Control sequence for Codablock F bar codes:

Parameter	Description
Cr*Rows*	Number of rows (2-44)
Cc*Columns*	Number of columns (4-62)
Ch*RowHeight*	Height of row in mm
Cs*SeperatorHeight*	Height of seperator in mm

Cf*Format*	Format (*Format* is taken from the table Codablock F Format)

Bar code position

TBarCode Filter supports two output formats: Postscript and PCL

In Postscript the origin of the coordinate system is the *lower left* corner of the page.

In PCL the origin of the coordinate system is the *upper left* corner of the page.

If no absolute position is given in the control sequence, the current position of the bar code control sequence is used as position for the barcode. This may cause unwanted results in some postscript documents. Therefore absolute positioning is recommended!

6 Web Applications

6.1 Generating Raster Images

The following steps demonstrate how to create bar code bitmap images with **TBarCode for UNIX**.

1. Generate a new bar code with **TBarCode Client**:
   ```
   tbarcodeclient -fbarcode.eps tPS b20 w80 h50 f24 O dDemo123
   ```

   This command creates a bar code with the following settings:
   -fbarcode.eps .....name of output file is "barcode.eps"
   tPS .....select postscript format for output
   b20 .....bar code type is Code128
   w80 .....width of bar code is 80mm
   h50 .....height of bar code is 50mm
   f24 .....font size is 24
   O .....width of the bar code is adjusted to fit into 72dpi matrix
   dDemo123.....data of bar code is "Demo123"

   If you intend to convert the EPS-file into a bitmap file it is important to set the width (e.g. w80) and the parameter O. Only in this way can the resulting EPS-file be converted to a raster format without the loss of information (narrow bar width will be adjusted to 72 dpi dot size). When parameter O is set, then it is likely that the resulting width is different than the specified width (80mm in the example above). This is unavoidable – but TBarCode for UNIX tries to use a width that is close to the specified width.

   Important: if you are using a constant module width (instead of setting the width directly with “w”), you should use a module width, which is equal to (or a multiple of) 0.353mm. Postscript uses a 72 dpi raster so 1 Pixel = 0.353. With this value for the module with you avoid raster errors.

2. Convert the EPS-file to bitmap format:
   The created file `barcode.eps` contains the bar code at a resolution of 72dpi.
   It can be converted to a raster file with the “convert” command line tool (http://www.imagemagick.com) or with Ghostscript.
   - Converting `barcode.eps` into a PNG-file using "convert":
     ```
     convert +antialias barcode.eps barcode.png
     convert +antialias barcode.eps barcode.jpg
     ```
     Please read the FAQ about convert if you have problems with aliasing (gray unsharp bars)
   - Converting barcode.eps into a PNG-file using Ghostscript:
     ```
     gs -dNOPAUSE -dBATCH -sDEVICE=pngmono -r72 -g225x143
        -sOutputFile=barcode.png barcode.eps
     ```
     (Parameter -g225x143 sets the size of the image. The size ("bounding box") can be determined width: `gs -dBATCH -sDEVICE=bbox barcode.eps`)

6.2 Dynamic Web Applications (PHP)

You can use TBarCode in dynamic web pages on your Linux server.

To create a bar code on demand in your PHP script, you can shell-execute “tbarcodeclient”:

- First you create a bar code EPS file
- Then convert it to a browser capable image file like JPG, GIF or PNG.

It is important that the module width of the bar code corresponds to the pixel size of the created image file (refer to previous chapter “Generating raster images”).

To display the bar code in the browser there are two possibilities:

1. Create the bar code image file with PHP and reference it in your html output:

```
// mypage.php – principle data flow
$tmp_bc_file = get_random_file_name();
$r = shell_execute (“tbarcodeclient … bar code parameters… $tmp_bc_file.eps”);
$r = shell_execute(“ghostscript … conversion parameters… ./imgpath/$tmp_bc_file”);
<html><body>
… <img src=”/imgpath/$tmp_bc_file”>
</body>
// after a while you need to delete the temporary created image files
// otherwise your hard drive will be flooded with bar code image files
```

2. Or reference a php script, which creates a bar code image data stream on demand

```
// mypage.php
<img src=”SendMeABarcode.php?data=1234”>

// SendMeABarcode.php  – principle data flow
header("Content-type: image/JPEG");
$unique_filename = dirname($PATH_TRANSLATED) . "\\" . "~" . uniqid(rand());
$r = shell_execute (“tbarcodeclient … $data… $unique_filename.eps”);
$r = shell_execute(“ghostscript … conversion parameters… . $unique_filename.jpg”);
// read the whole file and send it to the browser
$fp=fopen($unique_filename.”.jpg”,"rb");
fpassthru($fp); // pass through as binary data stream (JPG image format)
flush();
unlink ($unique_filename.”.eps”); // delete both files
unlink ($unique_filename.”.jpg”);
// make sure that you don’t add unwanted white space outside of the <?  ?> tags
```

Instead of the shell_execute() function you could also use exec() or system()

Hints for using shell_execute() (taken from php.net)

- If you're not getting any output from echo shellexec("prog") [for instance], at least try "./prog " before bothering with the full path.

- add '2>&1' to the end of your shell command to have STDERR returned as well as STDOUT. $shell_return = shell_exec($shell_command." 2>&1");
- Note: You cant used shell_exec() when safemode = on (its disabled), instead use exec() and copy the needed program into the /nonexec directory (by default, set in php.ini).
- When running subprocesses via shell_exec (and maybe others) from Apache/mod_php4, Apache's environment variables don't seem to be passed on to the subprocess environment unless you specifically force them by using putenv something like this:

```
$remaddr = getenv("REMOTE_ADDR");
putenv("REMOTE_ADDR=$remaddr");
shell_exec("/path/to/subprocess");
```


7 Licensing

7.1 License Key and License Types

TBarCode for UNIX can be used immediately after setup. As long as you have not licensed **TBarCode for UNIX** an additional horizontal bar will be printed over the bar code. Usually this horizontal bar does not affect the readability of the code for evaluation purposes.

The purchase of a license (and applying the license key) removes this restriction. There are three possible license modes to choose from:

- ➔ **Single**: This license gives you the right to use **TBarCode for UNIX** on exactly one computer (one client) and print to local printers. It is not permitted to use this license on a server or for network printers. Note: We need your System-ID for this license / refer to the section ***Troubleshooting/FAQ***.
- ➔ **Site**: This license gives you the right to use **TBarCode for UNIX** at exactly one site within your company, on as many clients as you want (also on a server and with network printers).
- ➔ **World** / **Multi-Site**: This license gives you the right to use **TBarCode for UNIX** worldwide at all sites of your company (no restrictions in the number of sites or clients).

Please note: Redistributing **TBarCode for UNIX** is generally NOT allowed – regardless of the license you purchased.

7.2 License File license.ini

After you have sent us your order, you will receive a special file with your registration data. This file is named "**license.ini**" and contains the license information and your license key.

Please copy this file into the directory of **TBarCode for UNIX** (`/usr/local/tbarcode`). If you are using a Site or a World License, you have to copy this file to each system (client) where you want to use the product. Overwriting the original (demo) license.ini file that was installed during setup has no unwanted effects.

After installing the license.ini file you have to restart the **TBarCode Daemon** if it is already running. Call `/usr/local/tbarcode/stoptbarcoded` (you need root privileges to do this). The daemon is started again automatically when you use either **TBarCode Client** or **TBarCode Filter**.

More information can be found on our web site http://www.tec-it.com/

8 APPENDIX

8.1 Barcode.ini

The file barcode.ini is the configuration file for **TBarCode Client** (not used by the filter). The section [CLIENT] contains following entries:

Entry	Description
Path_of_Project	Home directory of **TBarCode for UNIX**
Project_ID	Unique ID used for inter-process communication. (Changes should be made in steps of size 2).
Timeout	Timeout period in seconds
Connect_Retries	Number of retries to connect to **TBarCode Daemon**.
Use_Syslog	If this value is true daemon errors (and log messages) are reported using the syslog facility (syslogd).
Use_Stderr	If this value is true, daemon errors (and log messages) are printed to stderr.
Use_Error_File	If this value is true, daemon errors (and log messages) are written to an error file (if specified).
Path_of_Error_File	Full path of the error file
Path_of_Daemon	Location of tbarcoded
Size_Of_Shared_Memory	Size of shared memory in bytes
Debug	If this value is true, additional debug information is produced

Changes in barcode.ini should be made carefully.

If the daemon is running and changes are made to barcode.ini, the daemon should then be restarted to use the new settings.

Call `/usr/local/tbarcode/stoptbarcoded` (you need root privileges to do this). Daemon will be started automatically.

8.2 Bar code types

Number	Bar Code Type
0	Not a valid type
1	Code 11
2	Code 2 of 5 (Standard)
3	Interleaved 2 of 5 Standard
4	Code 2 of 5 IATA
5	Code 2 of 5 Matrix
6	Code 2 of 5 Data Logic
7	Code 2 of 5 Industrial
8	Code 3 of 9 (Code 39)
9	Code 3 of 9 (Code 39) ASCII
10	EAN8
11	EAN8 - 2 digits add on
12	EAN8 - 5 digits add on
13	EAN13
14	EAN13 - 2 digits add on
15	EAN13 - 5 digits add on
16	EAN128 (supports AIS)
17	UPC 12 Digits
18	CodaBar (2 width)
20	Code128
21	Deutsche Post Leitcode
22	Deutsche Post Identcode
25	Code 93
26	Identical to eBC_UPCA
29	RSS-14
33	UCC128 (= EAN128)
34	UPC A
35	UPC A - 2 digit add on

36	UPC A - 5 digit add on
37	UPC E
38	UPC E - 2 digit add on
39	UPC E - 5 digit add on
40	PostNet ZIP (5d.)
41	PostNet ZIP (5d.+CD)
42	PostNet ZIP (8d.)
43	PostNet ZIP+4 (5d.+4d.+CD)
44	PostNet DPBC (5d.+4d.+2d.)
45	PostNet DPBC (5d.+4d.+2d.+CD)
46	Plessey Code
47	MSI Code
48	SSCC18
50	LOGMARS
51	Pharmacode One-Track
52	Pharmazentralnummer
53	Pharmacode Two-Track
55	PDF417 - 2D bar code
56	PDF417 Truncated - 2D bar code
57	MaxiCode - 2D-bar code (Postscript only)
58	QR-Code
59	Code128 (CharSet A)
60	Code128 (CharSet B)
61	Code128 (CharSet C)
62	Code 93 Ascii
63	Australian Post Standard Customer
64	Australian Post Customer 2
65	Australian Post Customer 3
66	Australian Post Reply Paid
67	Australian Post Routing
68	Australian Post Redirection

69	ISBN Code (=EAN13P5)
70	Royal Mail 4 State (RM4SCC)
71	Data Matrix
72	EAN-14
74	Codablock-F
75	NVE-18

8.3 Check digit calculation method

Number	Check digit calculation method
0	No check digit calculation, no check digit added
1	The preset method of each bar code type will be used (that means also, that for some types no check digit is applied).
2	Modulo 10
3	Modulo 43 (suggested for Code39 and Logmars, consist of 1 digit)
4	Modulo 47 (2 check digits)
5	Methode for Deutsche Post Leitcode
6	Method for Deutsche Post Identcode
7	Method for Code 11 (1 check digit)
8	Method for Code 11 (2 check digits)
9	Method for USPS Postnet
10	Method for MSI (1 digit)
11	Method for MSI (2 digits)
12	Method for Plessey
13	Method for EAN 8
14	Method for EAN 13
15	Method for UPC A
16	Method for UPC E
17	Method for EAN 128
18	Method for Code 128
19	Method for Royal Mail 4 State

8.4 Data Matrix Parameters

Symbol Sizes

This table shows the possible user defined symbol sizes for ***Data Matrix***. The symbol size can be defined by the control sequence Ds*Index*

Set *Index* to the value, which corresponds to the selected size.

Index	Symbol size (rows x cols)	Index	Symbol size (rows x cols)
0	automatic calculation	16	64 x 64
1	10 x 10	17	72 x 72
2	12 x 12	18	80 x 80
3	14 x 14	19	88 x 88
4	16 x 16	20	96 x 96
5	18 x 18	21	104 x 104
6	20 x 20	22	120 x 120
7	22 x 22	23	132 x 132
8	24 x 24	24	144 x 144
9	26 x 26	25	8 x 18
10	32 x 32	26	8 x 32
11	36 x 36	27	12 x 26
12	40 x 40	28	12 x 36
13	44 x 44	29	16 x 36
14	48 x 48	30	16 x 48
15	52 x 52		

Format

This table shows the possible formats for ***Data Matrix***. The format can be defined by the control sequence Df*Index*

Set *Index* to the value, which corresponds to the selected format.

Index	Format
0	default format
1	UCC/EAN
2	Industry
3	Macro 05
4	Macro 06

8.5 Encoding bytes and control characters in input data

If you want to use non-printable or special characters in your bar code data, you have to use “Escape Sequences”. They always start with a backslash (‘\’) followed by the sequence (see table below). You can use them also for encoding binary data (Bytes) into your bar code, but only if the symbology offer this feature (e. g. PDF417 or Data Matrix).

Important - if you want to use escape sequences in your input data, put the data string into single quotation marks (like ‘123\F’) and enable translation of escape sequences with the “son” parameter

Implemented Escape Sequences:

Esc-Sequence	Description
\a	Bell (alert)
\b	Backspace
\f	Form feed
\n	New Line
\r	Carriage Return
\t	Horizontal Tab
\v	Vertical Tab
\\	The Backslash \ itself
\0ooo	ASCII-character in octal notation ooo octal digits (0..7)
\ddd	ASCII-character in decimal notation ddd decimal digits (0..9)
\xhh	For encoding Bytes or ASCII-characters in hexadecimal notation hh hexadecimal digits (0..F)
\F	FNC1 or Gs (\x1d), used in UCC/EAN codes as field separator
\E	ECI (**E**xtended **C**haracter **I**nterpretation), used in 2D codes like MaxiCode, Data Matrix and QR Code. Is used for switching between various code pages (multiple character sets) – contact us to get further information.
\EB, \EE	special ECI identifiers for nesting ECIs. \EB (**E**CI **B**egin) opens a nesting level, \EE (**E**CI **E**nd) closes it. Used in QR Code
\G	GLI (**G**lobal **L**anguage **I**dentifier), similar to ECI, but only used in PDF417.

Note: Please keep in mind, that when translation of escape sequences is enabled, you cannot code a backslash (“\”) directly. Use “\\” instead.

Encoding Bytes

With \xhh you can encode Bytes in hexadecimal notation, e.g. ‘\x01\xFF’ encodes the Byte 1 and 255.

Note: put the input data into single quotes, otherwise you need to encode a double-backslash \\ to get a single one.

Symbology specific control characters

If you have enabled translation of Escape sequences (parameter “son”) you can encode the following control characters (bar code type dependent).

Please note: the input data must contain the Escape sequence, which corresponds to the control character. Put the input data into single quotes (e.g. ‘123\210456’), otherwise you would need a double-backslash (like 123\\210456).

Escape sequence	Control character	Barcode type(s)
\210	FNC1	Code 128, EAN128, UCC128, 2D Codes
\211	FNC2	Code 128, EAN128, UCC128
\212	FNC3	Code 128, EAN128, UCC128
\213	FNC4	Code 128, EAN128, UCC128
\x11	DC1	Code93, Code93Ext
\x12	DC2	Code93, Code93Ext
\x13	DC3	Code93, Code93Ext
\x14	DC4	Code93, Code93Ext
\x1E	Rs	MaxiCode (Mode 3,4 SCM)
\x1D	Gs	MaxiCode (Mode 3,4 SCM)
\x04	Eot	MaxiCode (Mode 3,4 SCM)

8.6 Formatting bar code data

Format is like a ”picture” specification for formatting the bar code data prior to printing it (please do not mix up the *Format* with the *Ratio Format*) Placeholders in the specified format string can be mixed with constant data characters to build a final bar code data string. Also control characters are supported. With this feature it’s possible to:

- Select subsets in Code 128, EAN 128 and UCC 128 (even within the code!)
- Select the required start/stop character for CODABAR
- Change the position of the check digit
- Set the values of Date, Preamble, Service Class, Postal- and Countrycode directly in the barcode data (in conjunction with special Esc-sequences)

The placeholders are as follows:

Placeholder character	Description
#	Stands for the next data character
&	Stands for all remaining data characters
^	Stands for the next check digit (use only if check digits will be computed!)
A	Switch to Subset A (used in: Code 128, EAN 128, UCC 128)
B	Switch to Subset B (used in: Code 128, EAN 128, UCC 128)
C	Switch to Subset C (used in: Code 128, EAN 128, UCC 128)
A	Start- or stop character A (only in: CODABAR)
B	Start- or stop character B (only in: CODABAR)
C	Start- or stop character C (only in: CODABAR)
D	Start- or stop character D (only in: CODABAR)
S	Only for MaxiCode: enables setting the values of Date, Preamble, Service Class, Postal- and Country- Code directly in the barcode data (in conjunction with predefined Esc-sequences)

Examples:

Data input	Barcode type	Format string	Final bar code data used as content
123	Irrelevant		123
123	Irrelevant	5&	5123
123	Irrelevant	&6	1236
123	Irrelevant	q#w#e#	q1w2e3
123	Irrelevant	#q&	1q23
123	Irrelevant	&^	123c
123	Irrelevant	^&	c123
Hello	Code 128	A&	Hello
Hello	Code 128	A##B&	Hello
Hello4711	Code 128	A##B&	Hello4711
Hello4711	Code 128	A##B###C&	Hello4711

rote characters represented in Subset A
gray characters represented in Subset B
green characters represented in Subset C
c presents the place of the check digit

8.7 PCL Font Numbers

8.8 Use these font numbers in combination with the –f parameter (PCL output only).

Typeface Family	PCL Number
Albertus	4362
Antique Olive	4168
Claredon	4140
Coronet	4116
Courier	4099
Garamond Antiqua	4197
Letter Gothic	4102
Marigold	4297
CG Omega	4113
CG Times	4101
Univers	4148

8.9 MaxiCode Mode

This table shows the possible modes for ***MaxiCode***. The mode can be defined by the control sequence Mm*Index*

Set *Index* to the value, which corresponds to the selected format.

Index	Mode
2	SCM Numeric
3	SCM Alphanumeric
4	Default Mode
5	Full EEC

8.10 QRCode Parameters

Version (Symbol Sizes)

This table shows the possible user defined symbol sizes for ***QRCode***. The symbol size can be defined by the control sequence QRv*Index*

Set *Index* to the value, which corresponds to the selected size.

Index	Symbol size (rows x cols)	Index	Symbol size (rows x cols)
0	automatic calculation	21	101 x 101
1	21 x 21	22	105 x 105
2	25 x 25	23	109 x 109
3	29 x 29	24	113 x 113
4	33 x 33	25	117 x 117
5	37 x 37	26	121 x 121
6	41 x 41	27	125 x 125

7	45 x 45	28	129 x 129
8	49 x 49	29	133 x 133
9	53 x 53	30	137 x 137
10	57 x 57	31	141 x 141
11	61 x 61	32	145 x 145
12	65 x 65	33	149 x 149
13	69 x 69	34	153 x 153
14	73 x 73	35	157 x 157
15	77 x 77	36	161 x 161
16	81 x 81	37	165 x 165
17	85 x 85	38	169 x 169
18	89 x 89	39	173 x 173
19	93 x 93	40	177 x 177
20	97 x 97		

Format

This table shows the possible formats for ***QRCode*** barcodes. The format can be defined by the control sequence QRf*Index*

Set *Index* to the value, which corresponds to the selected format.

Index	Format
0	default format
1	UCC/EAN
2	Industry

Error Correction Level

This table shows the possible Error Correction Levels for ***QRCode*** barcodes. The Error Correction level can be defined by the control sequence QRe*Index*

Set *Index* to the value, which corresponds to the selected Level.

Index	Error Correction Level
0	Low
1	Medium
2	Quartil (Default)
3	High

8.11 Codablock F Parameters

Format

This table shows the possible formats for ***Codablock F*** barcodes. The format can be defined by the control sequence Cf*Index*

Set *Index* to the value, which corresponds to the selected format.

Index	Format
0	default format
1	UCC/EAN

8.12 Directory and files

The files of **TBarCode for UNIX** are located in the following directory:

```
/usr/local/tbarcode
```

Files in /usr/local/tbarcode:

Filename	Description
barcode.ini	Configuration file for **TBarCode Client**
license.ini	License file
stoptbarcoded	Script to stop **TBarCode Daemon** (You need root privileges to execute this script.)
tbarcodeclient	**TBarCode Client** executable
tbarcoded	**TBarCode Daemon** executable
tbarcodefilter	**TBarCode Filter** executable
tbarcodescript	Script to use **TBarCode Filter** with the LP system (see Installing TBarCode Filter)
tbarcodescript_fwd	Script to use **TBarCode Filter** with the LP system (see Installing TBarCode Filter)
testfile.ps	Postscript test-file for **TBarCode Filter**

8.13 Hex – ASCII Table

This table helps you to enter the Print Controls in Hex-Format. For each character exists an equivalent Hexcode – e.g. „C“ = Hex 43 or „2“ = Hex 32.

Hexcode	Zeichen	Hexcode	Zeichen	Hexcode	Zeichen	Hexcode	Zeichen
0	NUL	20	[space]	40	@	60	`
1	SOH	21	!	41	A	61	a
2	STX	22	"	42	B	62	b
3	ETX	23	#	43	C	63	c
4	EOT	24	$	44	D	64	d
5	ENQ	25	%	45	E	65	e
6	ACK	26	&	46	F	66	f
7	BEL	27	'	47	G	67	g
8	BS	28	(	48	H	68	h
9	HAT	29	)	49	I	69	i
A	LF	2A	*	4A	J	6A	j
B	VT	2B	+	4B	K	6B	k
C	FF	2C	,	4C	L	6C	l
D	CR	2D	-	4D	M	6D	m
E	SO	2E	.	4E	N	6E	n
F	ST	2F	/	4F	O	6F	o
10	SLE	30	0	50	P	70	p
11	CS1	31	1	51	Q	71	q
12	DC2	32	2	52	R	72	r
13	DC3	33	3	53	S	73	s
14	DC4	34	4	54	T	74	t
15	NAK	35	5	55	U	75	u
16	SYN	36	6	56	V	76	v
17	ETB	37	7	57	W	77	w
18	CAN	38	8	58	X	78	x
19	EM	39	9	59	Y	79	y
1A	STB	3A	:	5A	Z	7A	z
1B	ESC	3B	;	5B	[	7B	{
1C	FS	3C	<	5C	\	7C	|
1D	GS	3D	=	5D	]	7D	}
1E	RS	3E	>	5E	^	7E	~
1F	US	3F	?	5F	_	7F	□

8.14 Knowledge Base

Unix Printing (HP-UX and Solaris)

SVR4 spooling system

Solaris and HP-UX uses the SVR4 print services. Under SVR4 spooling system, the lp command accepts the data to be printed, makes a copy of it in the spool directory associated with the destination. The destination consists of a printer name and an optional specification of a class to which the printer belongs. When the specified printer is busy the job is sent to another printer in the same class. The spool directory is normally /var/spool/lp/request/printer-name and the print file is given a unique name to identify both the job and the user.

Access to the printer is controlled by lpsched daemon. It picks up the jobs from the spool directory and sends them to appropriate destination when it becomes available. lpsched also keeps a log, usually in /usr/spool/lp/log. The log file would indicate any error in processing the print jobs, as well as the user-name,

Interface Programs (BSD and SVR4)

Both BSD and SVR4 spooling systems support the concept of an interface program. The interface program, referred to as filters under the BSD system, is usually a shell script that translates the print file to a format suitable for the output device. The tasks performed by the interface program include: adding a banner and trailer pages, adding or removing a line feed character, generating accounting information and setting the correct modes on the output device. A standard interface program may be found in /usr/lib/lpf for the BSD systems and in /usr/spool/lp/model for the SVR4 system.

Printer Interface Scripts (HP-UX)

There are printer interface script "models" you can choose from that have been created for you in the /usr/spool/lp/model directory. Many of them have names that match the model numbers of Hewlett-Packard Printers and Plotters.

When you configure your printer into the lp spooler (e.g. with SAM), you must specify which printer model interface script you want to use. The model will be automatically copied from the /usr/spool/lp/model directory into the /usr/spool/lp/interface directory and given the name that you specified as printer name.

If you list the /usr/spool/lp/model directory you will find printer interface scripts like:

```
HPGL1, draftpro, hp2560, HPGL2, dumb, laserjet, PCL1, dumbplot, laserjetIIIS, PCL2, fonts,
hp2565a,hp33447a, paintjet, PCL3, hp2225a, hp2566b, hp3630a, quietjet, PCL4, hp2225d, hp2567b,
hp7440a, rmodel, PRINT3K.model, hp2227a,hp2631g,hp7475a,rmttroff, bf_remote, hp2228a, hp2684a,
hp7550a, ruggedwriter, colorpro, thinkjet, deskjet - many others.
```

If you have an HP printer, you will probably find a model script that matches its model number or name. Those interface model scripts that match your printers typically do not need to be changed - except we want to include TEC-IT bar code software.

In order to use "TBarCode Filter" we need some shell programming to customize the printer interface model scripts to meet our printing needs.

If you do not have an HP printer, try using the dumb interface model. You might have to modify it to be able to use all of the features of your non-HP printer, but "dumb" should work for basic ASCII text printing. If the dumb printer interface model script does not work, contact your printer supplier for a UNIX line printer spooler interface script or try the script that most closely matches your non-HP printer type.

Links

➔ Printing under Unix (BSD, SVR4…)
http://www.ussg.iu.edu/usail/peripherals/printers/

➔ Adding print queues (BSD, Solaris, IRIX, HP-UX…)
http://www.fisica.uniud.it/~cabras/assistenza/print/tek/man/P740man/74086.htm

➔ AIX/HP-UX Printing Guide / Interoperability
http://www.rz.uni-karlsruhe.de/Uni/RZ/Betriebssysteme/HP-UX/doc/interworks/Tech/aix_hpux_interop/chap08_print.html

8.15 Troubleshooting / FAQ

How can I verify that TBarCode for UNIX was installed successfully?

- Check if the files listed in Directory and files can be found in the directory */usr/local/tbarcoded*. Without these files bar code printing is not possible.
- Follow the instructions in Testing and using TBarCode Client and Testing and using TBarCode Filter.

How can I stop TBarCode Daemon?

Call the script stoptbarcoded: `/usr/local/tbarcode/stoptbarcoded`

(You need root privileges to execute this script.)

How to tell which filters are in place (and maybe failing?) or missing?

Switch on the "debug" mode in the LogLevel directive for your CUPS daemon. Edit /etc/cups/cupsd.conf (or wherever this configuration file is installed in your case) to have the line "LogLevel debug" there. Then restart your CUPS daemon /etc/software/init.d/cups start (the command may be customized by your Linux distribution; f.e. SuSE will take "rccups start").

Now print a job. Watch what is written to your CUPS error log. It normally sits in /var/log/cups/error_log. In debug level, nearly every action of the CUPS daemon is logged. You can see which filters and backends are called in which order. Very often you'll get a hint about what is missing for the print job to succeed.

More information about printing problems --> www.linuxprinting.org

How can I filter ASCII files?

Using TBarCode Filter has a special aspect: it is not possible to filter a raw ASCII text file directly; it must be PCL or Postscript printer language as input for the filter.

There are several ASCII-to-Postscript filters available (from the Linux/Unix vendors or third party), one of the common tools is “a2ps”. This filter must be called before the print job is passed to TBarCode Filter. Then TBarCode Filter adds the bar code according to the filter control sequences in the document.

If your printer has no Postscript capability, in most cases it can decode PCL Level 5 (very common, e.g. LaserJet 4/5). In this case the input to our filter must be PCL. Either your application create PCL input or you find an ASCII-to-PCL filter to do this.

One of the filter products, which converts ASCII to PCL on demand is Magicfilter. This program is standard for several Linux distributions and often installed in the spool system already. Using TBarCode Filter together with Magicfilter in order to generate PCL output can lead to problems. Postscript would be no problem.

How can I change the font family of the human readable text in PCL?

Font Height: f<size>

e.g. f12

Font Family: f<font-number>
e.g. f16602...Arial
f4099….Courier

If the number from the f parameter is 1000 or bigger than 1000, it will be identified as PCL-Font number.

More font numbers -> see appendix: PCL Font Numbers

I have problems with “convert” (gray bars inside the symbol).

The convert utility was originally made for image conversion (photographs) and has a built-in antialias filter. During conversion from 72 dpi EPS files to bitmap files this filter can produce unsharp bars with grayscales. There is an option called “+antialias” to switch off the filter but due to a bug this option may work or not.

Use the following workaround to get a clear image with convert:

The antialias filter doesn’t produce gray scaled bars if the resolution of the input file is big enough.

1. Create the bar code 4 times bigger than you need it:
   If you have a module width parameter of m353 use → m1411
   If you have a width parameter w100 use w400 (multiply your value with 4)
   If you have a height parameter of h20 use h80 (height * 4)
2. During conversion reduce the size to 25%:
   convert –scale 25\% barcode.eps barcode.png
3. Now you have the size you want and an image with clear content.

How can I generate a PDF417 bitmap?

PDF417 needs some specific settings to get optimum size:

1. the module width must be a multiple of 0.353 mms for 72 dpi EPS
2. for PDF417 the row height Ph should be >= 3 * Module width = 1.059 or can be also 5 * Module width = 1.765 mm

A sample call:

- tbarcodeclient -fbarcode.eps tPS b55 m353 Ph1.059 d"MyDataContent"
- now convert to JPG with: `convert barcode.eps barcode.jpg`
- “convert” is a freeware tool → www.imagemagick.org . This is the best method, because you don’t need to know the pixel dimensions of the EPS. With convert use the +antialias option to avoid gray bars. Also read our FAQ about convert problems.

Or use Ghostscript for image type conversion (EPS -> JPG/PNG/BMP…):

First you need to know the maximum (or real size) of the bar code in Pixels, then use this value for g

- gs -dNOPAUSE -dBATCH -sDEVICE=jpeg -r72 -g330x110 -sOutputFile=barcode.jpg barcode.eps

How can I encode an XML string with TBarCodeClient?

If you encode an XML string you need to take care of the quotes. It works only if you use the correct quotes (" or ') and removed the Carriage Returns.

Linux command line parameters can be either marked with double quotes or single quotes (if they contain a string). For XML encoding use single quotes to mark the data and then never use single quotes inside the data.

Example:

- tbarcodeclient -file.eps tPS b55 m706 Ph2.118 Pe3 d'<TED version="1.0"><DD><RE>78079790-8</RE><TD>33</TD><F>3602</F><FE>2003-10-20</F...'

Don't use single quotes inside the data, because Unix (Linux) stops the input string after the next ' character.

If you need to use single quotes ' or other special characters inside of your bar code data, switch Escape Sequences on with the "son" parameter and encode the special character with the following escape sequence: **xhh** (hh=hex Ascii code)

Example:

- tbarcodeclient -fname_image.eps ... son d'...Castaqa\x27s tostados\x27 CALIENTITOS...'
- Ascii \x27 = ' (single quote)

How to license the product: the "license.ini" file

After you have ordered **TBarCode for UNIX** you will receive your license key within the license.ini file. This file must be copied into the installation directory of **TBarCode for UNIX:** /usr/local/tbarcode. Then you have to restart the **TBarCode Daemon** if it is already running. Call /usr/local/tbarcode/stoptbarcoded. The daemon is automatically started again when you use either the **TBarCode Client** or the **TBarCode Filter**.

Once a valid license file is installed on your system, your restricted installation (with printing of the additional horizontal bar) becomes a full-featured installation without any restrictions.

The license data section in the license.ini File contains the following information:

- Product (1D or 2D)
  1D…linear bar codes: Code 2 of 5, Code 128, Code 39, EAN, UPC…
  2D…two-dimensional symbologies: PDF417, MaxiCode, Data Matrix, QR-Code
- Licensee (Company)
- License_Mode (Single/Site/World)
- Number_Of_Licenses
- License_Key

Why there is no barcode when I'm testing the filter under LPRng?

- the printdata have to include a barcode sequence – e.g.:
  $_tbcs tPCL b20 m254 h10.2 d0123456789$_tbce
- printing with lp (not lpr)
- the filter must be registered exactly in the printcap data (look at "Installation")

How to retrieve the hostname for buying a single license?

For a Single License we need the hostname of the computer (the client) where you want to use **TBarCode for UNIX**.

To get this hostname enter the following command at the command line (of the target system):

```
hostname
```

TBarCode Client reports that a shared library is missing

When starting **TBarcode Client** you receive the following error message:

```
Error when calling tbarcodeclient:
/usr/local/tbarcode/tbarcoded: error while loading shared libraries:
libtbarcode.so.1: cannot open shared object file: No such file or directory
tbarcodeclient: Couldn't find tbarcoded
```

Solution:

- Make sure that 'libtbarcode.so' is in '/usr/local/lib'. If it is missing, reinstall the libtbarcode-rpm-package.

- Run 'ldconfig /usr/local/lib' in your command line.

How can I debug TBarCode Filter?

To start **TBarCode Filter** and **TBarCode Daemon** in debug mode

- stop **TBarCode Daemon**: call `/usr/local/tbarcode/stoptbarcoded`
- call **TBarcode Filter** with parameter –x:

```
/usr/local/tbarcode/tbarcodefiler -x ...
```

How can I create a log-file with debug information from TBarCode Client?

In /usr/local/tbarcode/barcode.ini debug information can be activated with:

```
Debug=true
```

TBarCode Client will print debug information to the command line (stderr). To write the debug information to a file (e.g. error.txt) you can use output redirection, e.g.:

```
tbarcodeclient b20 dtest123 2>error.txt
```

If debug information of **TBarCode Daemon** should be sent to a file the following changes to /usr/local/tbarcode/barcode.ini should be made:

```
Use_Error_File=true
Path_of_Error_File=/tmp/tbarcoded.txt
```

Then the **TBarCode Daemon** should be stopped with:

```
/usr/local/tbarcode/stoptbarcoded
```

The next time **TBarCode Client** is called **TBarCode Daemon** will be restarted and debug information of the Daemon will be written to /tmp/tbarcoded.txt.

See Appendix Barcode.ini in this document for more information.

How can I create a log-file with debug information from TBarCode Filter?

Stop **TBarCode Daemon** with:

```
/usr/local/tbarcode/stoptbarcoded
```

To create a log-file you have to call **TBarCode Filter** with the command line parameter –f*path_of_logfile*.

To activate all debug information use the parameter –x. Output of debug information to the command line can be suppressed with the parameter –s.

e.g.

```
/usr/local/tbarcode/tbarcodefilter -f/tmp/log.txt -x -s ...
```

Debug information of **TBarCode Filter** will be written to /tmp/log.txt.

Debug information of **TBarCode Daemon** will be written to /tmp/log.txt.dmn.txt.

My printer queue/spooler reports an error although everything seems to work?

Older versions of TBarCode Filter (<=1.1-6) print messages if bar codes were generated successfully. Some spooler interpret these messages as error messages.
The messages can be suppressed by calling the filter with the parameter –s:

```
/usr/local/tbarcode/tbarcodefilter -s ....
```


TBarCode Client reports timeout error - what can I do?

If you are experiencing difficulties, please consider following hints:

1. **TBarCode Daemon** (`tbarcoded`) should not be killed with any signal other than the TERM signal! (Do not kill `tbarcoded` with the KILL signal.) Use the script `stoptbarcoded` to stop the daemon.
2. Following changes in the configuration file could help. Change the value in the barcode.ini file and restart the daemon afterwards:
   - Change the Project-ID by a number greater than 2. This causes **TBarCode Client** to start a new daemon. For example if the client does not get any response from the daemon then this will help.
   - Set Use_Stderr and Debug to true to enable additional debug output.
   - Increase the timeout value.
   - Increase the size of the shared memory block.
   - Increase the Connect_Retries value.
3. Control the file attributes. Following files must have execute rights and the User-ID (SUID) bit set:
   ```
   tbarcoded
   tbarcodeclient
   tbarcodefilter
   ```

   Missing attributes may be set (by root only) with: `chmod a+xs name_of_file`

   The following files need execute rights:
   ```
   tbarcodescript
   tbarcodescript_fwd
   tbarcodescript_cups
   ```

   Missing attributes may be set with: `chmod a+x name_of_file`

Where can I read syslog messages?

Syslog messages will be written to the appropriate file specified in /etc/syslog.conf.

Normally this is set to /var/log/messages.

Why is a horizontal bar drawn across the bar codes?

You are currently working with the restricted demo version. There is no valid or an invalid license file installed in the directory, where **TBarCode for UNIX** resides. Please refer to section Licensing or contact us for a valid license file.

How can I exchange my BarSIMM Escape sequences with tbarcode filter control sequences?

Example: Symbology 2of5 Interleaved

	PCL BarSIMM Sequence:	TBarCode Filter Sequence:
Barcode Prefix	**<Esc>(s1p37v24640T**	**$_tbcs tPCL b3 m254 h13 n ob d**
Barcode Suffix	**<Esc>(0N_(s3T**	**$_tbce**

Please note:

If you want to edit PCL print data directly (e.g. within a spool file during tests) please consider that a standard text editor could adulterate the print data during saving (umlauts, character set differences and CR/LF conversion). *Use a Hex Editor for PCL editing - e.g. the tool KHexEdit!*

We have tried to keep the default parameters for module width and symbol height as close the the BarSIMM default values as possible. Leave away the m and h parameter to get the predefined default values.

If you don't know, which bar code parameters are defined through your BarSIMM Escape Sequence(s), please send them to our support@tec-it.com - we try to find the corresponding TBarCode Filter parameters for you.

How can I print bar codes within a text file?

PCL printers can print not only PCL, they can print also normal ASCII data – therefore reports and list printing can be done with normal ASCII print files (and they are more often used as we think).

But when it comes to bar code printing, things will change. Now advanced drawing commands need to be added to the ASCII print job.

For TBarCode Filter this is no problem. Pass in your ASCII file (with filter control sequences) and call tbarcodefilter → in the output file you will have your ASCII print data extended with bar code drawing commands ready to be sent to a PCL printer.

But you need to send this file directly to the printer without going to the Linux standard spool filter (which is in most cases “magic filter”). Magic filter would process the print job again and would re-convert the bar code drawing commands back to text. Read the next FAQ for more information.

How can you send a file directly to a printer without being processed by magic filter or other tools?

This can be done with a remote queue. Your printer need to be setup up on a remote queue. In order to pass the file directly to this queue, use lpr with parameter –b

For example:

lpr -P PCLQueue@remotehost -b testfile.pcl

So first call tbarcodefilter with your text file:

/usr/local/tbarcode/tbarcodefilter <Textfile.txt >Textfile_with_barcode.pcl

And then print with *lpr -PHP4050PCL@karthago -b Textfile_with_barcode.pcl*

LPR Spool System: How can I find out which data the printer gets from the queue/spooler?

Stop the printer with „lpc stop“:

lpc stop PQ-D601 //PQ-D601 the queue called

While printing, a file with the printing data is in the directory.

For example:

Directory: /var/spool/lpd/PQ-D601

Postscriptfile: dfA2023...

Copy this data for analyze. Then start the printer again:

lpc

start PQ-D601

Now the spool files will be processed and no longer shown.

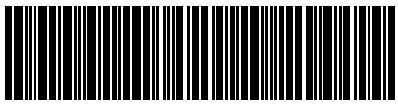
Where I can get more help?

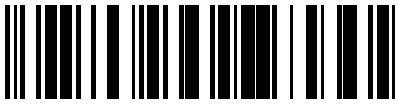
We do our best to support our customers – please contact us:

TEC-IT Datenverarbeitung GmbH
Wagnerstraße 6
A-4400 Steyr (Austria)

Phone: +43/7252/72720
Fax: +43/7252/72720-77
www.tec-it.com
mailto:sap@tec-it.com

8.16 Supported bar codes

2 of 5 Industrial		
Valid characters:	“0”..”9”	0123456789
Check digit method:	Modulo10	
Default:	no check digit	
Notes:		

2 of 5 Interleaved		
Valid characters:	“0”..”9”	0123456789
Check digit method:	Modulo10	
Default:	no check digit	
Notes:	If the number of digits is odd a leading zero will be inserted automatically. This bar code type can encode only an even number of digits	

2 of 5 Matrix		
Valid characters:	“0”..”9”	0123456789
Check digit method:	Modulo10	
Default:	no check digit	
Notes:		

8.16.1.1.1.1 CodaBar		
Valid characters:	“0”..”9”, “-“, “$”, “:”, “/”, ”.”, ”+”, “A”, “B”, “C”, “D”	A01234:/.+ A
Check digit method:	-	
Default:	no check digit	
Notes:	“A”, “B”, “C”, “D” are only useable as start or stop characters.	

8.16.1.1.1.2 Code 39		
Valid characters:	“0”..”9”, “A”..”Z”, “-“, “.”, “ “, “*”, “$”, “/”, “+”, “%”	ABC123-+
Check digit method:	Modulo43	
Default:	no check digit	
Notes:	Start- and stop characters (*) are created automatically and must not be included in the input data.	

8.16.1.1.1.3 Code 39 Extended		
Valid characters:	ASCII-characters between 0..127	ABCabc()?
Check digit method:	Modulo43	
Default:	no check digit	
Notes:	Start- and stop characters (*) are created automatically and must not be included in the input data.	

8.16.1.1.1.4 Code 93		
Valid characters:	“0”..”9”, “A”..”Z”, “-“, “.”, “ “, “$”, “/”, “+”, “%”	ABC123-/+ 2J
Check digit method:	Modulo47 (2 digits)	
Default:	Modulo47	
Notes:	Start- and stop characters (*) are created automatically and must not need be included in the input data.	

8.16.1.1.1.5 Code 93 Extended		
Valid characters:	ASCII-characters between 0..127	ABCabc123-/+ T0
Check digit method:	Modulo47 (2digits)	
Default:	Modulo47	
Notes:	Start- and stop characters (*) are created automatically and must not need be included in the input data.	

8.16.1.1.1.6 Code 128 Subset B		
Valid characters:	ASCII-characters between 0..127	ABCabc123-/+
Check digit method:	Check digit included in the code	
Default:	-	
Notes:		

8.16.1.1.1.7 EAN 8		
Valid characters:	“0”..”9”, 7 digits + 1 Check digit	9031 1017
Check digit method:	Check digit included in the code	
Default:	-	
Notes:	Check digit is automatically calculated if not in the input data (that is when only 7 digits are used for creating the code). Used for article bar coding.	

8.16.1.1.1.8 EAN 8 with 2 digits add-on	
Valid characters:	“0”..”9”
Check digit method:	Check digit included in the code
Default:	-
Notes:	Same as EAN8, but with 2 add-on digits enclosed

8.16.1.1.1.9 EAN 8 with 5 digits add-on	
Valid characters:	“0”..”9”
Check digit method:	Check digit included in the code
Default:	-
Notes:	Same as EAN8, but with 5 add-on digits enclosed

8.16.1.1.1.10 EAN 13	
Valid characters:	“0”..”9”, 12 digits + 1 Check digit
Check digit method:	Check digit included in the code
Default:	-
Notes:	Check digit is automatically calculated if not in the input data (that is when only 12 digits are used for creating the code). Used for article bar coding.

8.16.1.1.1.11 EAN 13 with 2 digits add-on	
Valid characters:	“0”..”9”
Check digit method:	Check digit included in the code
Default:	-
Notes:	Same as EAN13, but with 2 add-on digits enclosed.

8.16.1.1.1.12 EAN 13 with 5 digits add-on	
Valid characters:	“0”..”9”
Check digit method:	Check digit included in the code
Default:	-
Notes:	Same as EAN13, but with 5 add-on digits enclosed.

8.16.1.1.1.13 UPC version A	
Valid characters:	“0”..”9”, 11 digits + 1 Check digit
Check digit method:	Check digit included in the code
Default:	-
Notes:	Check digit is automatically calculated if not in the input data (that is when only 11 digits are used for creating the code). Used for article bar coding.

8.16.1.1.1.14 UPC version A, 2 digits add-on		
Valid characters:	"0"..."9"	
Check digit method:	Check digit included in the code	
Default:	-	
Notes:	Same as UPC version A, but with 2 add-on digits enclosed.	

8.16.1.1.1.15 UPC version A, 5 digits add-on		
Valid characters:	"0"..."9"	
Check digit method:	Check digit included in the code	
Default:	-	
Notes:	Same as UPC version A, but with 5 add-on digits enclosed.	

8.16.1.1.1.16 UPC version E		
Valid characters:	"0"..."9", 7 digits + 1 Check digit	
Check digit method:	Check digit included in the code	
Default:	-	
Notes:	Check digit is created automatically when not committed in the input data (that is when only 7 digits are used for creating the code). Used for article bar coding. Code must begin with "0" or "1".	

8.16.1.1.1.17 UPC version E, 2 digits add-on		
Valid digits:	"0"..."9"	
Check digit method:	Check digit included in the code	
Default:	-	
Notes:	Same as UPC version E, but with 2 add-on digits enclosed.	

8.16.1.1.1.18 UPC version E, 5 digits add-on		
Valid digits:	"0"..."9"	
Check digit method:	Check digit included in the code	
Default:	-	
Notes:	Same as UPC version E, but with 5 add-on digits enclosed.	

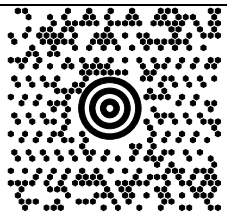
8.16.1.1.1.19 UCC / EAN-128		
Valid characters:	ASCII characters between 0..127	
Check digit method:	Check digit included in the code	
Default:	-	
Notes:	Standardized version of Code 128.	

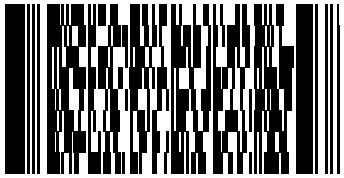
8.16.1.1.1.20 SSCC18		**Equivalent to EAN128**
Valid characters:	ASCII-characters between 0..127	
Check digit method:	Check digit included in the code	
Default:	-	
Notes:	Special edition of EAN128	

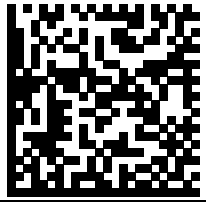
8.16.1.1.1.21 MSI		**8.16.1.1.2 Supported, but without check digits**
Valid characters:	“0”..”9”	
Check digit methods:	Not implemented	
Default:	No check digit	
Notes:		

8.16.1.1.2.1 USPS Postnet 5		
Valid characters:	“0”..”9”, 5 digits + 1 Check digit	
Check digit method:	Check digit included in the code	
Default:	-	
Notes:	Check digit is calculated automatically if not posted in the input data (that is when only 5 digits are used for creating the code). Used for postal purposes.	

8.16.1.1.2.2 USPS Postnet 9		
Valid characters:	“0”..”9”, 9 digits + 1 Check digit	
Check digit method:	Check digit included in the code	
Default:	-	
Notes:	Check digit is calculated automatically if not posted in the input data (that is when only 9 digits are used for creating the code). Used for postal purposes.	

8.16.1.1.2.3 MaxiCode (2D Symbology)		
Valid characters:	Alphanumeric and/or numeric	
Check digit method:	Check digit and error correction included in the code	
Mode:	Mode-4 (standard symbol)	
Notes:	Used by UPS. Modes for including postal information (SCM) can be adjusted. Printing size is set to a norm value.	

8.16.1.1.2.4 PDF-417 & PDF417 Trunc (2D Symbology)		
Valid characters:	alphanumeric and/or bytes	
Check digit method:	check digit and error correction included in the code	
Notes:	2D symbology (multi-row) to encode larger quantities of data. Data representation is divided into rows and columns that adjust automatically (depending on input data) or can be set by printer commands. Also available as truncated version: PDF417 Trunc	

8.16.1.1.2.5 Data Matrix (2D Symbology)		
Valid characters:	alphanumeric and/or bytes	
Check digit method:	check digit and error correction included in the code	
Notes:	Powerful 2D symbology to encode larger quantities of data. Size adjusts automatically depending on input data.	

New symbologies will be added continuously. Not listed symbologies are available on request.

Available in upcoming versions:

- → EAN UCC RSS Reduced Space Symbology
- → EAN UCC Composite Codes
- → Japanese Postal Code
- → Korean Postal Authority Code
- → Planet Bar code
- → Codablock F

