INTERBUS
Conformance Test

PCP Test

Table of Contents

1 Introduction

This part of the specification for the INTERBUS conformance test describes the general test environment for the server, client, and profile tests. The emphasis is on the system for testing INTERBUS-Peripherals-Communication-Protocol devices (PCP devices). A description is given of the structure and generation of the PICS file, the formation of the test result (conclusion) for a test case, and the reference system of the test case. The test cases themselves are not referred to here. Special cases or amendments (in particular those encountered while testing devices without PCP) are referred to in the special documentation for the individual test classes.

2 Test Objective

The aim of this test is to test the behavior of an INTERBUS test object (INTERBUS device) submitted by a manufacturer, with regard to a specific property. In this case the behavior with regard to its function as an INTERBUS device (server), an INTERBUS Peripherals Communication Protocol device (server and/or client) is evaluated. Particular attention is paid to the behavior of the test object, which can be seen at the respective interface (INTERBUS driver or PCP-ALI).

PCP device behavior is stipulated in the INTERBUS PCP reference manual. The test system is responsible for examining the test object and testing its reaction, from the point of view of conformity with the reference manual or the respective profile specification, using the PCP service primitives. During a server test, the application software currently running on the device is used. For the client test, supporting test software (‘Upper-Tester’) is necessary on the device side. A test application is available for the PCP server testing of PC slave interface boards, and can be installed on the PC.

3 Test Environment

3.1 Test and Measurement Tools

The test and measurement tools required for carrying out the conformance test are listed in Appendix A of part 1 "General". The PCP test software is essential for carrying out the test.

3.2 Manufacturer Specific Data

In order to carry out the conformance test, the manufacturer of the test object must include a floppy disk in MS-DOS format, which contains either the PICS file (the format is described below) or the three files kbl.dat, vfd.dat and ov.dat (from which the PICS file can be generated). The three *.DAT files are required for implementation of the manufacturer's protocol software.

A help tool is available for generating the PICS file for PCP 2.0 implementation. This can be used by the manufacturer. Implementation errors can be recognized at an early stage using the PICS generator.

3.3 Test Configuration

Figures 1 and 2 indicate the test configuration with regard to hardware and Figures 3 to 5 with regard to software. Two PCs are required for testing lateral communication. In parallel to the test system software, the PTED Phoenix monitor program must be active on the second PC. After starting this program, the PTED command 'w' should then be entered to automatically load the CRL (Auto_Load_KBL). The Phoenix Contact INTERBUS PC AT-T board is currently used as the interface board in the PC.

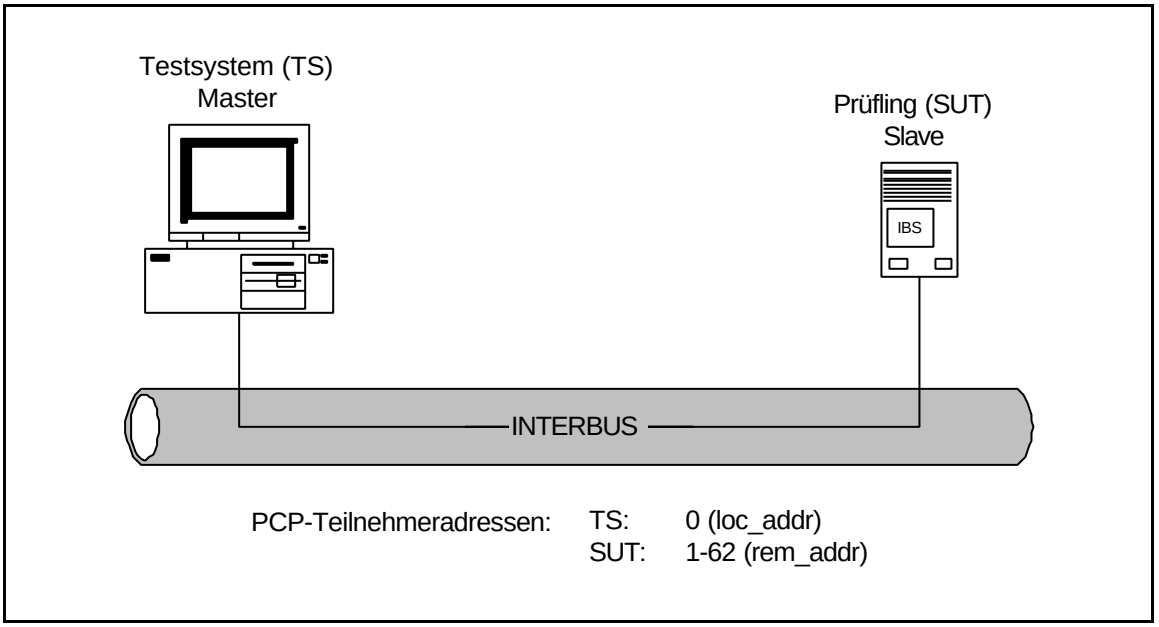

Figure 1: Test configuration (hardware) for testing without lateral communication

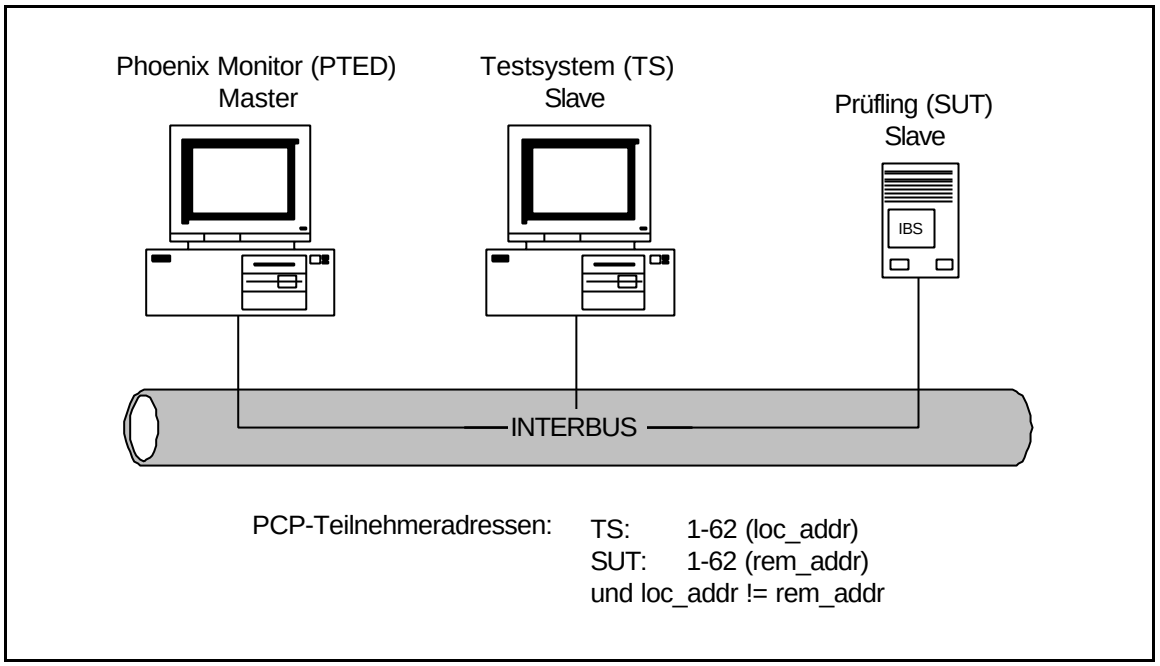

Figure 2: Test configuration (hardware) for testing with lateral communication

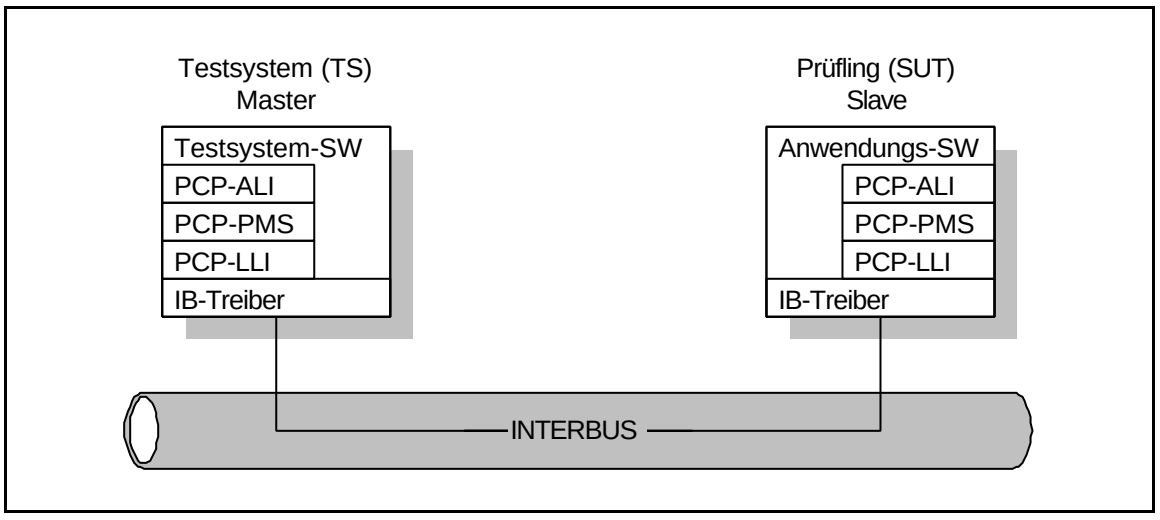

Figure 3: Server test (for devices with PCP (address))

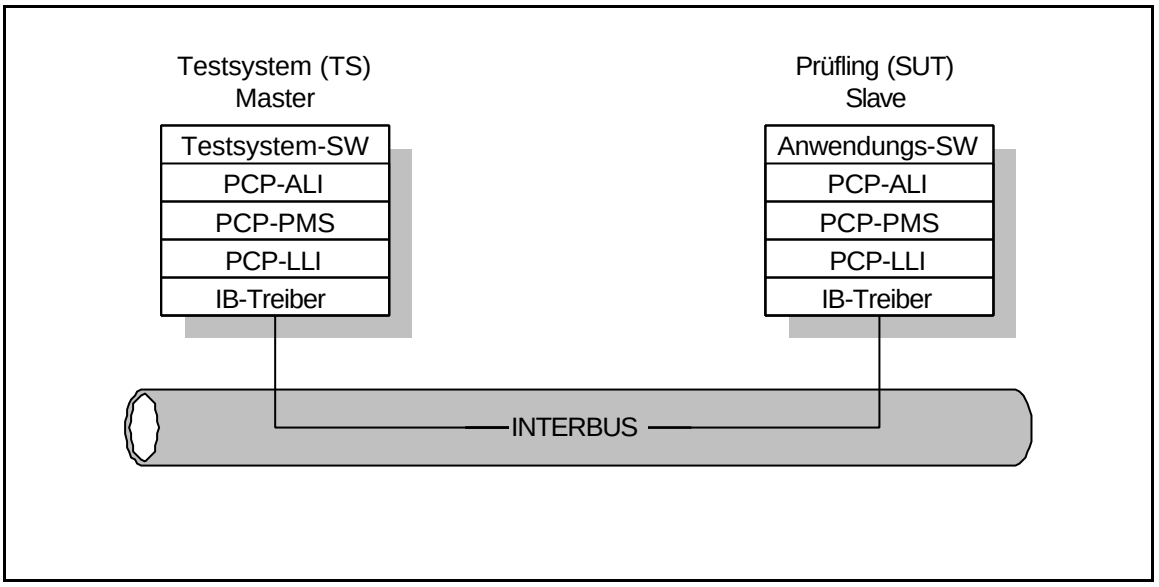

Figure 4: Client test (only for devices with PCP (address))

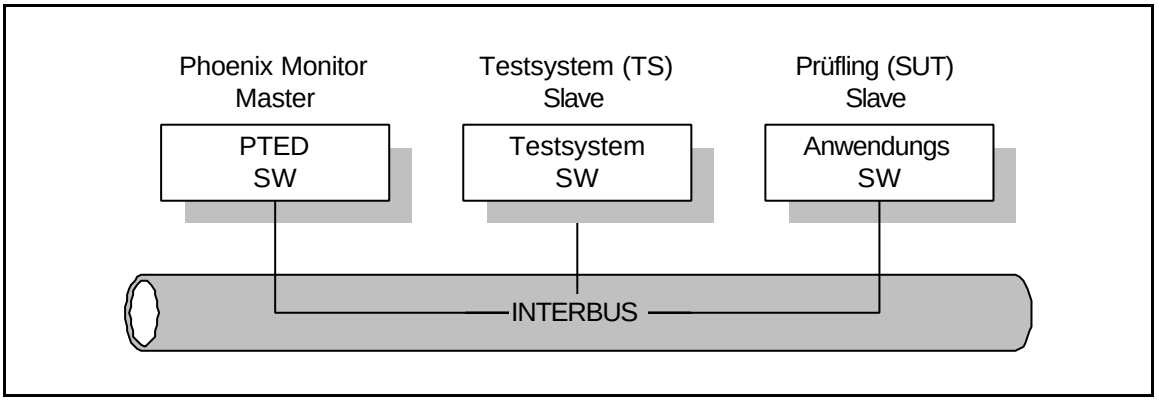

Figure 5: Lateral communication test (only for devices with PCP (address))

It should be noted that:

– In the PCP test, the test system software makes direct use of the PCP-ALI (Application Layer Interface) which considerably reduces test system programming efforts. This means that the behavior of the test object can only be recognized by the ALI programming interface of the test system. The degree of testing for "normal", correct operation is not influenced in the case of fault-free reference implementation for the test system. However, the generation of faults for testing the fault detection and reconfiguration mechanisms at the lower levels is restricted.

– The application software is used as an ‘Upper-Tester’ during the server test, i.e. the services specified in the PCP reference manual are used for to the objects implemented here.

Advantages of testing without an ‘Upper-Tester’:

– The device manufacturer saves time and memory space required for installing or adapting the ‘Upper-Tester’.

– The application is incorporated in the test.

Disadvantages:

– The test is only complete with regard to the implemented services, objects etc.

– Problems with accessing the objects are not excluded, as the object semantics are not known to the test system (for help see Section 4.1.1.3 Test Restrictions).

4 PICS/PIXIT File

The test specifications allow the device manufacturer a great deal of freedom during implementation. For example, there are:

– mandatory and optional services

– mandatory and optional objects

– parameters which are freely adjustable within certain limits

In principle, the test system must be structured in such a way that all specified properties can be tested. However, when testing a single device, only the following may be tested:

– presence of all mandatory properties

– compliance with the specifications for all mandatory properties and optional properties which have also been implemented

It is therefore necessary to give the test system an exact description of the test object and the properties to be tested at the beginning of the test. This is achieved by means of the PICS/PIXIT file.

4.1 Syntax and Semantics

4.1.1 Devices With PCP

4.1.1.1 Server and Client Test for Devices With PCP

For the server and client test, the PICS/PIXIT file contains all PCP-relevant details relating to the test object, in two parts.

The first part contains details of the test configuration and procedure:

– Details of the PCP version to be tested (the current PCP version will always be tested).

– The PCP address at which the test object can be found.

– A monitoring time in seconds, which specifies the maximum length of time the test system will wait for the service to be executed in the test object.

The second part contains a description of the object properties as implemented by the manufacturer, which are required for creating the following PCP data structures and all data structures contained and/or referred to therein.

– T
  _KBL_HDR user_kbl_hdr (CRL header)
– T
  _OV_OBJ_DESCR_HDR ov_obj_descr_hdr (Object dictionary)
– T
  _VFD_OBJECT vfd (VFD object)
– T
  _KBL_ENTRY_STAT user_kbl[] (Communication relationships)

The descriptions appear in the PICS/PIXIT file in four main sections, each of which starts with the keyword "PICS_PART:". The sections are as follows:

– OTHER,
– CRL,
– OD and
– VFD.

With the exception of OTHER, these sections may occur in any order. They contain sub-sections whose order is fixed. The sub-sections start and terminate with keywords. All four main sections (OTHER, VFD, OD and CRL) must be included in the PICS/PIXIT file and may not occur more than once.

When the input format was determined, it was ensured that the names for the attributes, the data types, the enumeration types and the order in which the attribute values are to be specified are correspondingly found in the INTERBUS PCP reference manual and/or the INTERBUS PCP programming manual. For this reason, no explanation is given below of the meaning of the various names and their reference to PCP. To ensure full comprehension, it is essential to study the PCP manuals.

4.1.1.1.1 PICS_PARTs Formats and Attribute Description

The PICS/PIXIT file may contain an unlimited number of comment lines. Comment lines are identified by the "*" character in the first column.

Description format:

```
Name of attribute     Data type              Meaning
*
PICS_PART: OTHER                             * keyword
*--------------------------
*
* All other data
*
---other---                                  * keyword
simulation:           BOOL                       * only FALSE permitted
loc_addr:             USIGN8                 * PCP address of
                                             * test system
rem_pms_version:      STRING                 * PMS version to be tested
rem_add_qual:         T_ADDR                 * optional: see test class
rem_addr:             USIGN8                 * PCP address of test object
rem_time_out:         USIGN8                 * max. waiting time for
                                             * response in seconds
---end---                                    * keyword
*
*
PICS_PART: VFD                               * keyword
*----------------------
*
* VFD
*
---vfd---                                    * keyword
vendor_name:          STRINGV                * T_VFD_IDENT
model_name:           STRINGV
revision:             STRINGV
profile_name:         STRING8
logical_status:       T__LOGICAL_STATUS      * T_VFD_STATUS
physical_status:      T__PHYSICAL_STATUS
local_detail:         STRING8
---end---                                    * keyword
*
*
PICS_PART: KBL                               * keyword
*----------------------
*
* Communication relationship list header
*
```

```
---hdr_kbl---                                   * keyword
com_ref:            T_COMREF
size:               USIGN8
poll_listen_ssap:   USIGN8
ass_abt_ci:         USIGN32
symbol_length:      USIGN8
vfd_ptr_supp:       BOOL
---end---                                       * keyword
*
* Communication relationship list
*
---usr_kbl---                                   * keyword
*
* Number of following blocks = size
*
com_ref:            T_COMREF
local_lsap:         USIGN8
rem_addr:           USIGN8
remote_lsap:        USIGN8
conn_type:          T__CONN_TYPE
lli_sap:            USIGN8
conn_attr:          T__CONN_ATTR
max_scc:            USIGN8
max_rcc:            USIGN8
max_sac:            USIGN8
max_rac:            USIGN8
aci:                USIGN32
multiplier:         USIGN8
req_len_h:          USIGN8
req_len_l:          USIGN8
ind_len_h:          USIGN8
ind_len_l:          USIGN8
serv_sup_client:    T__PMS_SERVICE or T__PNM7_SERVICE
serv_sup_server:    T__PMS_SERVICE or T__PNM7_SERVICE
symbol:             STRINGV
vfd_pointer:        USIGN32
*
---end---                                       * keyword
*
*
* In the case of default management connection and
* support of remote CRL loading:
*
---load_kbl_rem_params---
```

```
*
max_crl_size:          USIGN8
pms_serv_sup_client:   T__PMS_SERVICE
pms_serv_sup_server:   T__PMS_SERVICE
pnm7_serv_sup_client:  T__PNM7_SERVICE
*
---end---
*
*
PICS_PART: OV                                  * keyword
*---------------------
*
* Object dictionary header
*
---hdr_ov---                                  * keyword
index:                 USIGN16
flag:                  BOOL
length:                USIGN8
protection:            BOOL
version:               USIGN16
len_st_ov:             USIGN16
first_index_s_ov:      USIGN16
len_s_ov:              USIGN16
first_index_dv_ov:     USIGN16
len_dv_ov:             USIGN16
first_index_p_ov:      USIGN16
len_p_ov:              USIGN16
---end---                                     * keyword
*
*
* Object dictionary
*
* All null objects may be implemented explicitly and implicitly. * For explicitly
implemented null objects
* (obj_code=NULL_OBJECT), the following applies:
*
index:                 USIGN16
obj_code:              T__OBJ_CODE
*
*
* Standard data types
*
---r_ov---                                    * keyword
*
length:                USIGN16                 * number of following blocks
```

```
 *
* For implemented standard data types (obj_code=DATA_TYPE_OBJECT)
*
index:              USIGN16
obj_code:           T__OBJ_CODE
meaning:            STRINGV
*
---end---                                   * keyword
*
*
* Non-standard data types and data structure types
*
---st_ov---                                 * keyword
*
length:             USIGN16                 * number of following blocks
*
* For objects with obj_code = DATA_TYPE_OBJECT
*
index:              USIGN16
obj_code:           T__OBJ_CODE
meaning:            STRINGV
*
* For objects with obj_code = TYPE_STRUCT_OBJECT
*
index:              USIGN16
obj_code:           T__OBJ_CODE
no_of_elements:     USIGN8
*
index_of_type:      USIGN16                 * no_of_elements blocks
length:             USIGN8
.
.
index_of_type:      USIGN16
length:             USIGN8
*
---end---                                   * keyword
*
*
* Objects of the static object dictionary
*
* Objects in the static object dictionary may be specified
* with a range object description if they directly adjoin one
* another and have the same properties. This also applies
* to null objects. The index data is replaced with range data
```

```
* with index_first and index_last:
*
* index_first:          USIGN16
* index_last:           USIGN16
* obj_code:             T__OBJ_CODE
* .
* .
*
* By specifying the access right W_USER_DATA, it is
* possible to specify data which is used for write access to
* the object being tested. Details are given under user_data
* at the end of the object description:
*
* .
* .
* extension:            STRING8
* user_data:            STRING8
*
*
---s_ov---                                    * keyword
*
length:                 USIGN16               * number of following blocks
*
* For objects with obj_code = SIMPLE_VAR_OBJECT
*
index:                  USIGN16
obj_code:               T__OBJ_CODE
index_of_type:          USIGN16
length:                 USIGN8
pass_word:              T_PASSWORD
acc_groups:             T__ACCESSGROUPS
acc_right:              T__ACCESSRIGHTS_V
symbol:                 STRINGV
extension:              STRING8
*
* For objects with obj_code = STRING_VAR_OBJECT
*
index:                  USIGN16
obj_code:               T__OBJ_CODE
index_of_type:          USIGN16
length:                 USIGN8
pass_word:              T_PASSWORD
acc_groups:             T__ACCESSGROUPS
acc_right:              T__ACCESSRIGHTS_V
symbol:                 STRINGV
```

```
extension:          STRING8
*
* For objects with obj_code = ARRAY_OBJECT
*
index:              USIGN16
obj_code:           T__OBJ_CODE
index_of_type:      USIGN16
length:             USIGN8
nof_elements:       USIGN8
pass_word:          T_PASSWORD
acc_groups:         T__ACCESSGROUPS
acc_right:          T__ACCESSRIGHTS_V
symbol:             STRINGV
extension:          STRING8
*
* For objects with obj_code = RECORD_OBJECT
*
index:              USIGN16
obj_code:           T__OBJ_CODE
index_of_type:      USIGN16
pass_word:          T_PASSWORD
acc_groups:         T__ACCESSGROUPS
acc_right:          T__ACCESSRIGHTS_V
symbol:             STRINGV
extension:          STRING8
*
---end---                                   * keyword
*
*
* Objects of the dynamic variable list dictionary
*
---dv_ov---                                 * keyword
*
length:             USIGN16                 * number of following blocks
*
* For objects with obj_code = VARIABLE_LIST_OBJECT
*
index:              USIGN16
obj_code:           T__OBJ_CODE
no_of_elements:     USIGN8
index:              USIGN16                 * no_of_elements index values
.
.
index:              USIGN16
```

```
pass_word:          T_PASSWORD
acc_groups:         T__ACCESSGROUPS
acc_right:          T__ACCESSRIGHTS_V
deletable           BOOL
symbol:             STRINGV
extension:          STRING8
*
---end---                                   * keyword
*
*
* Objects of the PI dictionary
*
---p_ov---                                  * keyword
*
length:             USIGN16                 * number of following blocks
*
* For objects with obj_code = INVOCATION_OBJECT
*
index:              USIGN16
obj_code:           T__OBJ_CODE
no_of_domains:      USIGN8
pass_word:          T_PASSWORD
acc_groups:         T__ACCESSGROUPS
acc_right:          T__ACCESSRIGHTS_PI
deletable:          BOOL
reusable:           BOOL
symbol:             STRINGV
extension:          STRING8
*
---end---                                   * keyword
```

In accordance with the designated type, the values for the attributes must be described as follows:

USIGN8,
USIGN16,
T_PASSWORD,
T_COMREF: Hexadecimal number. The prefix 0x is omitted.
STRING: Character string.
STRINGV: Character string length <character string>. Characters which cannot be represented are described by \NNN, where NNN is a three-digit decimal number in the range The character \ is described by \\.

STRING8: Character string length <character string>. The individual characters are represented as two-digit hexadecimal numbers, separated by a blank. The prefix 0x is omitted.

For STRINGV and STRING8, the specified character string length must correspond to the number of characters which are delimited by the characters < and >. The character string length is to be specified as a decimal number.

BOOL,
T__OBJ_CODE,
T__ACCESSGROUPS,
T__ACCESSRIGHTS_V,
T__ACCESSRIGHTS_PI,
T__CONN_TYPE,
T__CONN_ATTR,
T__LOGICAL_STATUS,
T__PHYSICAL_STATUS,
T__SERVICE: Enumeration type Several permissible alternatives are separated by a comma.

The following ranges of values are defined:

USIGN8: 0x00 <= value <= 0xff.

USIGN16: 0x00 <= value <= 0xffff.

STRING: Character string consisting of printable characters (without blanks).

STRINGV: Character string consisting of numbers in the range 0-127.

STRING8: Character string consisting of hexadecimal numbers in the range 0x00-0xff.

BOOL = {TRUE, FALSE}.

T__OBJ_CODE = {NULL_OBJECT, OV_OBJECT, DATA_TYPE_OBJECT, TYPE_STRUCT_OBJECT, SIMPLE_VAR_OBJECT, STRING_VAR_OBJECT, ARRAY_OBJECT, RECORD_OBJECT, VARIABLE_LIST_OBJECT, INVOCATION_OBJECT}.

T__ACCESSGROUPS = {1, 2, 3, 4, 5, 6, 7, 8, -}:

1	= access group 1
2	= access group 2
3	= access group 3
4	= access group 4
5	= access group 5
6	= access group 6
7	= access group 7
8	= access group 8
-	= no access group

T__ACCESSRIGHTS_V = {R, W, Rg, Wg, Ra, Wa, NO_R, NO_W, NO_RW, W_USER_DATA, -}:

R	= see PCP reference manual
W	= see PCP reference manual
Rg	= see PCP reference manual
Wg	= see PCP reference manual
Ra	= see PCP reference manual
Wa	= see PCP reference manual
NO_R	= no read access to test object permitted
NO_W	= no write access to test object permitted
NO_RW	= no access to test object permitted
W_USER_DATA	= write access with manufacturer-specific data
-	= there are no access rights

T__ACCESSRIGHTS_PI = {S, H, Sg, Hg, Sa, Ha, NO_SH, H_USER, -}:

S	= see PCP reference manual
H	= see PCP reference manual
Sg	= see PCP reference manual
Hg	= see PCP reference manual
Sa	= see PCP reference manual
Ha	= see PCP reference manual
NO_SH	= no starting/stopping of a PI permitted during the test
H_USER	= application influences PI state machine in PI_RUNNING state (transition to PI_IDLE or PI_STOPPED)
-	= there are no access rights

T__CONN_TYPE = {MMAZ}.

T__CONN_ATTR = {CONN_ATTR_D,CONN_ATTR_O}.

T__LOGICAL_STATUS = {
L_STAT_KOMMUN_BEREIT, L_STAT_BEGR_ANZ_SVC,
L_STAT_READY_FOR_COMM, L_STAT_LIMITED_NO_SVC}.

T__PHSICAL_STATUS = {
P_STAT_BETRIEBS_BEREIT, P_STAT_TEILW_BEREIT,
P_STAT_NICHT_BEREIT, P_STAT_WARTUNG_ERFORDERL,
P_STAT_READY_FOR_OP, P_STAT_PARTIALLY_READY,
P_STAT_NOT_READY_FOR_OP, P_STAT_SERVICE_REQUIRED}.

T__PMS_SERVICE = {MANDATORY, INFO_REP, GET_OV_LONG, PI,
READ, WRITE}.
MANDATORY = all Mandatory Services are supported
INFO_REP = the Information Report Service is supported
GET_OV_LONG = the Get OD Service is supported in the long version
PI = the Start/Stop Service is supported
READ = the Read Service is supported
WRITE = the Write Service is supported
T__PNM7_SERVICE = {MANDATORY, LOAD_KBL_REM,
READ_KBL_REM, -}
MANDATORY = all Mandatory Services are supported
LOAD_KBL_REM = the Load CRL Rem Service is supported
READ_KBL_REM = the Read CRL Rem Service is supported

4.1.1.1.2 PICS_PARTs Formats and Attribute Description

The PICS/PIXIT file may contain an unlimited number of comment lines. Comment lines are identified by the "*" character in the first column.

Description format:

Name of attribute	Data type	Meaning
*		
PICS_PART: OTHER_PROFILE		* keyword
*----------		
*		
* Identification header for profile		
*		
---other---		* keyword
prof_name:	STRING	* profile name
prof_version:	STRING	* profile version
prof_number:	STRING8	* profile number
---end---		* keyword
*		
*		

```
PICS_PART: OV_OBJ_PROFILE                    * keyword
*-------------------------------------------
*
* The object description is identical to the description under
 * PICS_PART OV in 4.1.1.1.1
*
*
PICS_PART: OV_PAR_PROFILE                    * keyword
*-------------------------------------------
*
* Value description and object function
*
* Standard data types
*
---r_ov---                                   * keyword
*
length:             USIGN16                  * number of following blocks
*
index:              USIGN16                  * object index
class:              T__CLASS                 * object class
*
---end---
*
*
* Data structure types
*
---st_ov---                                  * keyword
*
length:                                      * number of following blocks
*
index:              USIGN16                  * object index
class:              T__CLASS                 * object class
no_of_elem_to_supp: USIGN8                   * number of elements
                                             * to be supported *
*
---end---                                    * keyword
*
*
* Objects of the static object dictionary
*
---s_ov---                                   * keyword
*
length:             USIGN16                  * number of following blocks
*
* For objects with obj_code = NULL_OBJECT
```

```
*
index:                  USIGN16                 * object index
class:                  T__CLASS                * object class
*
* For objects with obj_code = SIMPLE_VAR_OBJECT
*
index:                  USIGN16                 * object index
class:                  T__CLASS                * object class
process:                T__PROCESS              * process data image
error_codes:            T__ERROR_CODE           * error codes
range:                  T__RANGE                * profile-specific range of *
values
range_to_supp:          T__RANGE                * profile-specific range of
mandatory range
default:                T__VAL                  * substitute value
*
* For objects with obj_code = ARRAY_OBJECT or RECORD_OBJECT
 *
index:                  USIGN16                 * object index
class:                  T__CLASS                * object class
process:                T__PROCESS              * process data image
error_codes:            T__ERROR_CODE           * error codes
no_of_elem_to_supp: USIGN8                      * number of elements
                                                * to be supported
*
* Description of the elements
*
subindex:               USIGN8                  * sub-index
range:                  T__RANGE                * profile-specific range of *
values
range_to_supp:          T__RANGE                * profile-specific range of
mandatory range
default:                T__VAL                  * substitute value
*
*
---end---                                       * keyword
*
*
* Objects of the dynamic variable list dictionary
*
---dv_ov---                                     * keyword
*
length: 0
*
```

```
---end---                                   * keyword
*
*
* Objects of the PI dictionary
*
---p_ov---                                  * keyword
*
length: 0
*
---end---                                   * keyword
*
*
*
PICS_PART: OV_PAR_DICS                      * keyword
*-------------------------------------
*
* Profile-independent, profile-specific properties of the test object
*
* Objects of the static object dictionary
*
---s_ov---                                  * keyword
*
length:             USIGN16                 * number of following blocks
*
* For objects with obj_code = SIMPLE_VAR_OBJECT
*
index:              USIGN16                 * object index
range_supp:         T__RANGE                * manufacturer-specific * range
of values
*
*
* For objects with obj_code = ARRAY_OBJECT or RECORD_OBJECT
 *
index:              USIGN16                 * object index
no_of_elem_supp:    USIGN8                  * number of elements
                                            * to be supported.
*
* Description of the elements
*
subindex:           USIGN8                  * sub-index
range_supp:         T__RANGE                * manufacturer-specific * range
of values
*
*
---end---                                   * keyword
```

```
*
*
* Objects of the dynamic variable list dictionary
*
---dv_ov---                                   * keyword
*
length: 0
*
---end---                                     * keyword
*
*
* Objects of the PI dictionary
*
---p_ov---                                    * keyword
*
length: 0
*
---end---                                     * keyword
*
*
*
```

In accordance with the designated type, the values for the attributes must be described as follows:

USIGN8,	
USIGN16,	
USIGN32,	
INT16:	Hexadecimal number. The prefix 0x is omitted.
STRING, STRING8:	See 4.1.1.1.1
T__CLASS,	
T__PROCESS:	Enumeration type
T__ERROR_CODE:	Number_of_Error Codes{Error Codes}
T__RANGE:	Number_of_Values range{Values}
T__VAL:	Default value

The following ranges of values are defined:
USIGN8, USIGN16, STRING, STRING8: See 4.1.1.1.1

USIGN32:	0x00000000 <= value <= 0xffffffff.
INT16:	0x8000 <= value <= 0x7fff.

T__CLASS = {OPTIONAL_OBJECT, MANDATORY_OBJECT}:
OPTIONAL_OBJECT = optional object (may be implemented),
MANDATORY_OBJECT = mandatory object (must be implemented).

T_PROCESS = {PE, PA, PEA, -}:
PE = image of process input data possible,
PA = image of process output data possible,
PEA = image of process data for input and output data possible,
- = no process data image possible.

T_RANGE:
Ranges to which no values have been assigned:
EQ_DEF = range corresponds to the definition/value range,
NO_DEF = there is no further restriction.

Ranges to which a value has been assigned:
EQ_VAL 1{value1} = only value 1 is supported,
NE_VAL 1{value1} = all values except value 1 are supported,
IN_MASK 1{value1} = all bits within the mask with value 1 are supported.

Ranges to which two values have been assigned:
IN_RANGE 2{value1, value2} = values with value1<=value<=value2 are supported.

The following types of ranges are used to specify the ranges of values:

Specification of the profile-specific range of values (range):
EQ_DEF: Range of values = definition range of the implemented data type. NE_VAL: Range of values as in EQ_DEF, but without the specified value.

Specification of the profile-specific mandatory range (range_to_supp):
NO_DEF: There is no mandatory range. EQ_VAL: The value must be supported. IN_RANGE: The specified range must be supported. IN_MASK: The bits within the mask must be supported (only for data type OCTET_STRING).

Specification of the manufacturer-specific range of values (range_supp):
EQ_DEF: Manufacturer-specific range of values = profile-specific range of values. EQ_VAL: Only this value and a mandatory value specified with

EQ_VAL are supported.
IN_RANGE:
The specified range and a mandatory value specified with EQ_VAL are supported. IN_MASK:
The bits within the mask are supported
(only for data type OCTET_STRING).

4.1.1.2 Test Restrictions

In some applications, it may be necessary for access to certain implemented objects to be possible only to a limited extent or, in extreme cases, even prohibited during the test. The creator of the PICS/PIXIT file is therefore given the opportunity of restricting the access rights (‘acc_right’) for the test (reading/writing an object of the static object dictionary, starting/stopping a PI (Program Invocation)) before the PICS/PIXIT file is accepted into the test system, without influencing the rights defined in the PMS. To put it more simply: depending on the PMS access right, for example, it is permissible to start a PI, yet the restriction by the creator of the PICS/PIXIT file ensures that this right is not exercised during the test. However, the restrictions do not apply to tests during which invalid access is to be rejected.

Besides the access rights defined in the PMS (R, W, Rg, Wg, Ra, Wa, S, H, Sg, Hg, Sa, Ha), the following additional restrictions are permitted:

For access to objects of the static object dictionary:
NO_W: Write access not permitted. NO_R:
Read access not permitted. NO_RW:
Read/write access not permitted. W_USER_DATA:
Write access only permitted using the
manufacturer-specific data.

For access to objects of the PI dictionary:
NO_SH: Starting/stopping of a PI not permitted. H_USER:
Application influences PI status machine in
PI_RUNNING state
(transition to PI_IDLE or PI_STOPPED)

The NO_xx test restrictions should only be used occasionally, as their use may lead to an "inconclusive" test result.

When executing the Write Service, the creator of the PICS file is able to specify the write buffer data for objects of the static object dictionary. This data is then used during execution of the Write Service for these objects by means of variable lists.

It is possible to specify range object descriptions for the static object dictionary. These can be used if objects immediately adjoining one another have the same properties (except for the index and symbol attributes). During the test, with the exception of the Get OD test in the start index, only the first object will then be used.

4.2 Generation

Figure 6 shows how PICS/PIXIT files are generated for testing devices with PCP. PICS/PIXIT files for testing devices without PCP may be generated by the device manufacturer using an editor. Sample data files simplify the generation process.

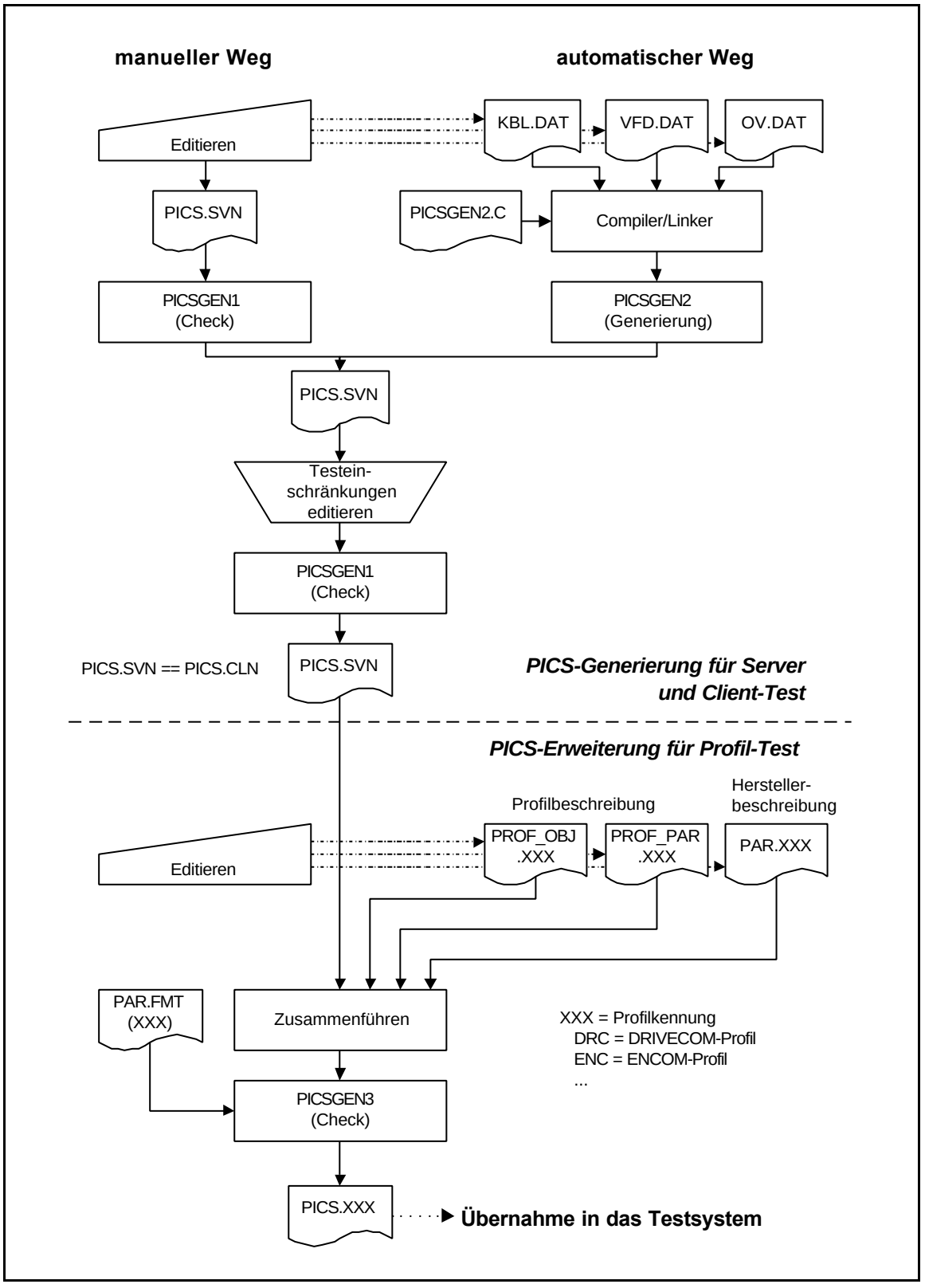

Figure 6: PICS/PIXIT file generation

4.2.1 Generation for the Server and Client Test

The PICS/PIXIT files ‘pics.svn’ and ‘pics.cln’ are of identical structure. There are two methods of generating these files:

– manual generation with an editor and

– automatic generation using the ‘kbl.dat’, ‘vfd.dat’ and ‘ov.dat’ C source files of the test object, provided they have been generated in accordance with the requirements of the PCP programming manual. If the ‘.dat’ files refer to variables which have not been defined therein, it is necessary to prepare these files in advance.

In the case of manual generation, the ‘pics.svn’ file is generated with the help of an editor e.g., Norton Editor, in the required PICS/PIXIT format. The ‘picsgen1’ program is used to check the generated file for completeness, observance of the syntax rules, consistency and PCP conformity.

In the case of automatic generation, the ‘kbl.dat’, ‘vfd.dat’ and ‘ov.dat’ files, which have usually already been generated by the manufacturer, are used as include files for the ‘picsgen2.c’ C source file and used to generate the ‘picsgen2’ program. When this program is executed, the desired conversion to the PICS/PIXIT format takes place and the result is the ‘pics.svn’ file.

The ‘pics.svn’ file can be processed further for the server test, for test restrictions according to the rules in Section 4.1.1.3.

For the data in PICS_PART OTHER, default settings are used which the creator of the PICS/PIXIT file must check and correct, if necessary. The default settings are as follows:

```
simulation:          FALSE          * only FALSE permitted
loc_addr:            0
rem_pms_version:     2.0            * current PCP version
rem_addr_qual:       PMS_ADDR
rem_addr:            1              * PCP address of test object
rem_time_out:        1              * max. waiting time for
                                    * response in seconds
```

The keyword ‘rem_addr_qual’ is used to further define the access procedure. The address of the ‘Upper Tester Agent’ object, for example, is recorded here for the client test. The keyword may be omitted if no further qualification is necessary for access. In this case, the address specified under ‘rem_addr’ will automatically be the PCP address. A description is given in the documentation for the relevant test class.

4.2.2 Generation Procedure

For the automatic generation of PICS/PIXIT files, the ‘picsgen’ command is used with the following syntax:

picsgen <pics_path> <class>

<pics_path>:	Path to the directory with the PICS/PIXIT files
<class>:	Test class to which the PICS/PIXIT files belong

Example: picsgen a:\pics drc (generation for DRIVECOM profile).

Generation takes place at various levels:

– If the ‘pics.drc’ file exists in the specified directory, only ‘picsgen3’ is executed.

– If the ‘pics.drc’ file does not exist and the ‘pics.svn’ file does exist, the ‘pics.drc’ file is generated from the ‘pics.svn’, ‘prof_obj.drc’, ‘prof_par.drc’ and ‘par.drc’ files.

– If the ‘pics.svn’ and ‘pics.drc’ files do not exist, ‘picsgen2’ is executed and the ‘pics.svn’ file is first constructed from the ‘kbl.dat’, ‘vfd.dat’ and ‘ov.dat’ files, followed by generation of the ‘pics.drc’ file.

4.3 Validation

4.3.1 Validation in the Server and Client Test

When the PICS/PIXIT file ‘pics.svn’ is read in, it is checked for

– completeness
– correct syntax (see 4.1)
– consistency
– version conformity (in this instance to PCP Version V2.0)

4.3.1.1 Consistency in the Server and Client Test

When the PICS/PIXIT file is read in, the following is checked in the sub-sections:

kbl_hdr:

com_ref	=	0	
size	<	128	
symbol_length	<	SYMBOL_LEN (12).	

usr_kbl:

com_ref:	If a default management connection is present, this must begin with 1 or 2 and then continue in an uninterrupted sequence.
rem_addr:	Each address may only occur once within the PMS or PNM7-CRL.

hdr_ov:

index	=	0	
length	<	SYMBOL_LEN	
len_st_ov	>=	0xe	
first_index_s_ov	>	len_st_ov	**
first_index_dv_ov	>=	first_index_s_ov + len_s_ov	**
first_index_p_ov	>=	first_index_dv_ov + len_dv_ov	**
first_index_p_ov + len_p_ov - 1 <= 0xffff			**

**: The tests are only carried out if the object ranges are present (length != 0).

ov:

index and index_of_type:	In each case, the values must be within their respective ranges.
index:	Must be in ascending order.
index_of_type:	References to null objects are not permitted. (obj_code = NULL_OBJECT or null object not explicitly implemented.
length:	The length data for the real objects based on PMS standard data types must correspond to the length data given in the PMS reference manual. For VISIBLE_STRING, OCTET_STRING, and BIT_STRING object types, the length must be > zero.

r_ov:

obj_code:	May only take the values DATA_TYPE_OBJECT or NULL_OBJECT.

st_ov:

obj_code:	May only take the values DATA_TYPE_OBJECT, TYPE_STRUCT_OBJECT or NULL_OBJECT,
index_of_type:	Must contain a reference to the data type.

s_ov:

obj_code:	May only take the values SIMPLE_VAR_OBJECT, STRING_VAR_OBJECT, ARRAY_OBJECT, RECORD_OBJECT or or NULL_OBJECT.
index_of_type:	For objects with obj_code = RECORD_OBJECT, it must contain a reference to the data structure type, or the data type.

dv_ov:

obj_code:	May only take the values VARIABLE_LIST_OBJECT or NULL_OBJECT.
index:	May only contain a reference to

SIMPLE_VAR_OBJECT,

ARRAY_OBJECT or
RECORD_OBJECT objects.

p_ov:

obj_code:	May only take the values INVOCATION_OBJECT or NULL_OBJECT.

s_ov/p_ov:
Within the individual object dictionary domains, each name may only be used once, provided that symbolic addressing is possible. Names with zero length are not included in the comparison.

If the Rg, Wg, Sg or Hg access right is set, at least one access group must be specified. When allocating R, W, S and H access rights, the password must not be left blank.

For program invocations, at least one of the access rights S, Sg or Sa must be set for an operational access rights test.

4.3.1.2 Version Conformity in the Server and Client Test

The following attributes are tested for conformity to PCP Version V2.0:

hdr_kbl:

poll_listen_ssap	=	NIL,
ass_abt_ci	<=	65535
symbol_length	<=	0xb,
vfd_ptr_supp	=	FALSE.

usr_kbl:
General:

local_lsap	=	NIL,		
remote_lsap	=	NIL,		
conn_type	=	MMAZ,		
aci	=	0		or
aci	=	0xffffffff,		
multiplier	=	NIL,		
req_len_h	=	0,		
ind_len_h	=	0,		
vfd_pointer	=	0.		

For default management connection (com_ref = 1):

lli_sap	=	1,		
conn_attr	=	CONN_ATTR_O		
max_scc	=	0,		
max_rcc	=	1,		
max rac	=	0,		
max_sac	=	0,		
53	>=	req_len_l	<=	246,
53	>=	ind_len_l	<=	246,
serv_sup[0]	=	0.		

For PMS connection:

lli_sap	=	0,		
conn_attr	=	CONN_ATTR_D		
max_scc	=	1,		
max_rcc	=	1,		
max rac	=	1,		
max_sac	=	1,		
51	>=	req_len_l	<=	246,
51	>=	ind_len_l	<=	246.

For PNM7 connection (com_ref != 1):

lli_sap	=	1,		
conn_attr	=	CONN_ATTR_D		
max_scc	=	1,		
max_rcc	=	0,		
max rac	=	0,		
max_sac	=	0,		
53	>=	req_len_l	<=	246,
53	>=	ind_len_l	<=	246,
serv_sup[1]	=	0.		

vfd:
profile_name[0] = 2.

hdr_ov:
length <= 0xb.

ov:
dv objects:
deletable = FALSE

p objects:
no_of_domains = 0,
deletable = FALSE,
reusable = TRUE.

5 Test Cases

The task of the test system is to test the behavior of an INTERBUS test object (device) with regard to a specific property, its function as an INTERBUS device (server and/or client) with and without PCP and/or its behavior as a member of a specific application class (profile). For this purpose, the individual test cases to be tested are stipulated in a test specification. The test cases stipulate how a test object is to behave under certain conditions.

5.1 Concepts

The test cases are mapped in the form of test programs used to test the individual properties of the test object. Test cases, which taken overall serve to test a higher-level property, are combined to form test classes. Examples of the implemented test classes are:

– Test class SVN for the PCP server function test.

– Test class CLN for the PCP client function test.

– Test class DRC for the function test according to the DRIVECOM profile specification.

– Test class ENC for the function test according to the ENCOM profile specification.

Within a test class, the test cases may be combined to form test schedules. When executing a test schedule, the test cases are then executed in the order in which they are included. In test classes SVN and CLN, test cases which test the same services are combined to form test schedules. The test schedule ‘all’ generally contains all the test cases of a single test class. Error-free execution of this is required for attainment of the certificate for this test class.

5.2 Classification System

For the naming of the test cases, the classification according to Figure 7 and the following explanations are used:

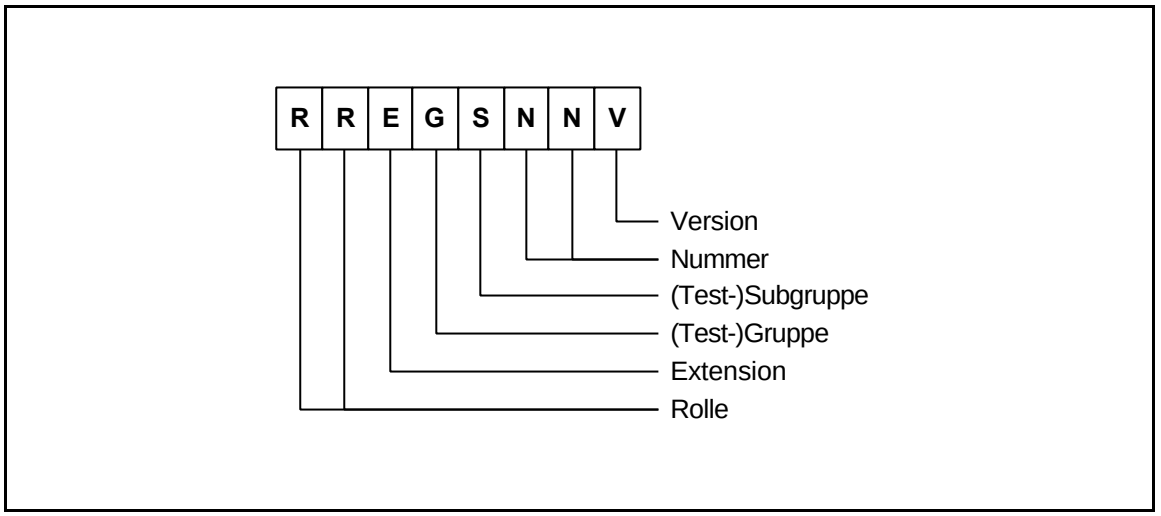

Figure 7: Naming of the test cases

Explanation of Figure 7:

The following applies to the server and client test:

Role: SV = PCP server function test of a test object
CL = PCP client function test of a test object
Extension: E = extended test, i.e. test via level 2 (not currently implemented)
N = normal test, i.e. test via PCP/ALI connection

The following applies to the profile test:

Role: DRC = DRIVECOM profile test
ENC = ENCOM profile test
...

For the profile test, more details may be given in the respective profiles; the following details apply to the server and client test:

CHECK test cases

Group:
0 = Implementation
PMS Sub-group:
0 = Object Dictionary
1 = PMS services objects
2 = PMS services objects buffer lengths
3 = Objects object attributes
PNM7 sub-group:
6 = Communication relationship list (CRL)
7 = PNM7 services CRL

PMS test cases

Group:

- 1 = Context management
  - Sub-group:
    - 0 = Initiate service (compatible PMS/LLI context)
    - 1 = Initiate service (incompatible ")
    - 2 = Abort service
    - 3 = Reject service
    - 4 = State machine
- 2 = VFD support
  - 0 = Identify service
  - 1 = State service
- 3 = OD management
  - 0 = Get OD service (mandatory)
  - 1 = Get OD service long (optional)
- 4 = Program invocation management
  - 0 = Start service
  - 1 = Stop service
  - 2 = Resume service
  - 3 = Reset service
  - 4 = State machine
  - 5 = State machine after power-on
- 5 = Variable access
  - 0 = Read service (Static OD)
  - 1 = Write service (Static OD
  - 2 = Read service (Dyn. variable list OD)
  - 3 = Write service (Dyn. variable list OD)
  - 4 = Information report service

PNM7 test cases

Group:

- 6 = Context management
- 7 = Configuration management

PMS-PNM7 test cases

Group:

- 8 = PMS and PNM7 services

A test case with the designation SVN10000 therefore belongs to the SVN test case class, i.e. it tests the function of a server via the PCP-ALI interface. It also belongs to group 1 and sub-group 0, i.e. it forms part of the context management conformance test, here specifically the initiate service.

6 Test Procedure

6.1 Adapting the Test System to the Test Object

Since the specifications allow the device manufacturer a great deal of freedom during implementation, it must be possible to adapt the test cases to the device which is actually to be tested. The devices must always fulfill the mandatory functions and be tested for them; optional functions may be fulfilled and need only be tested in this instance. The degrees of freedom for the parameters must be taken into account. The test case (and/or the creator of the test case) gathers this data from knowledge of the specification and from the device description in the PICS/PIXIT file, mapped on the 'pics' data structure. Before starting a test case, the communication relationship list (CRL), the object dictionary (OD) and the VFD object in the test system must be adapted to the test object and/or the test case, if applicable.

6.1.1 Communication Relationship List

The starting point for all PCP tests is that, at the beginning of the test, the communication references in the communication relationship list of the test system form a "mirror image" of the communication references in the communication relationship list of the test object (Figure 8).

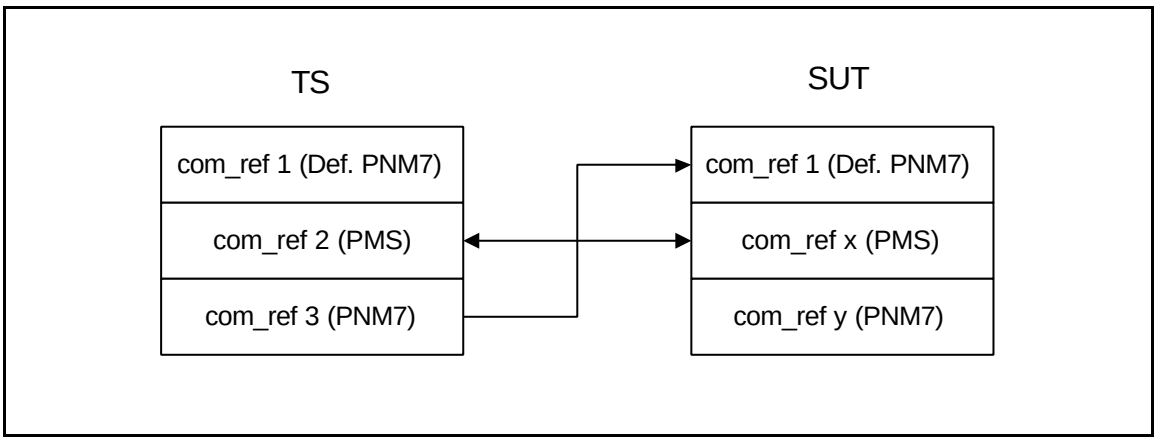

Figure 8: Relationships between the communication references of the test system and the test object

In what follows, the "remote" attribute always refers to the test object and the "local" attribute to the test system.

The structure of the communication references in the test system is as follows:

com_ref 0:

size	=	3	
poll_listen_ssap	=	NIL	
ass_abt_ci	=	0.9	* rem_time_out
symbol_length	=	0	
vfd_ptr_supp	=	0	

com_ref 1:

local_lsap	=	NIL
rem_addr	=	ALL
remote_lsap	=	NIL
conn_type	=	MMAZ
lli_sap	=	1
conn_attr	=	CONN_ATTR_O
max_scc	=	0
max_rcc	=	1
max_sac	=	0
max_rac	=	0
aci	=	0
multiplier	=	NIL
req_len_h	=	0
req_len_l	=	53
ind_len_h	=	0
ind_len_l	=	246
serv_sup	=	optional services are not supported
symbol	=	not used
vfd_pointer	=	0

com_ref 2:

local_lsap	=	NIL
rem_addr	=	PCP address of the test object
remote_lsap	=	NIL
conn_type	=	MMAZ
lli_sap	=	0
conn_attr	=	CONN_ATTR_D
max_scc	=	1
max_rcc	=	1
max_sac	=	1
max_rac	=	1
aci	=	aci (remote)
multiplier	=	NIL
req_len_h	=	0

req_len_l	=	ind_len_l (remote)
ind_len_h	=	0
ind_len_l	=	req_len_l (remote)
serv_sup	=	remote .req/.cnf becomes local .ind/.res remote .ind/.res becomes local .req/.cnf
symbol	=	not used
vfd_pointer	=	0

com_ref 3:

local_lsap	=	NIL
rem_addr	=	PCP address of the test object
remote_lsap	=	NIL
conn_type	=	MMAZ
lli_sap	=	1
conn_attr	=	CONN_ATTR_D
max_scc	=	1
max_rcc	=	0
max_sac	=	0
max_rac	=	0
aci	=	aci (remote)
multiplier	=	NIL
req_len_h	=	0
req_len_l	=	ind_len_l (remote)
ind_len_h	=	0
ind_len_l	=	req_len_l (remote)
serv_sup	=	remote .req/.cnf becomes local .ind/.res remote .ind/.res becomes local .req/.cnf
symbol	=	not used
vfd_pointer	=	0

This mapping ensures compatibility of the local PMS and LLI context with the remote PMS and LLI context (the ideal case is shown here).

6.1.2 Object Dictionary

The local object dictionary only needs to be adapted to the respective CLN test case for the client test. The adaptation serves to provide the appropriate PMS objects for testing the requests of the test object as client.

6.1.3 VFD Object

The local VFD object is only changed in the client test for test pattern generation for the VFD services.

6.2 Test Case Handling

Figure 9 shows the main procedural framework within which the test cases are incorporated.

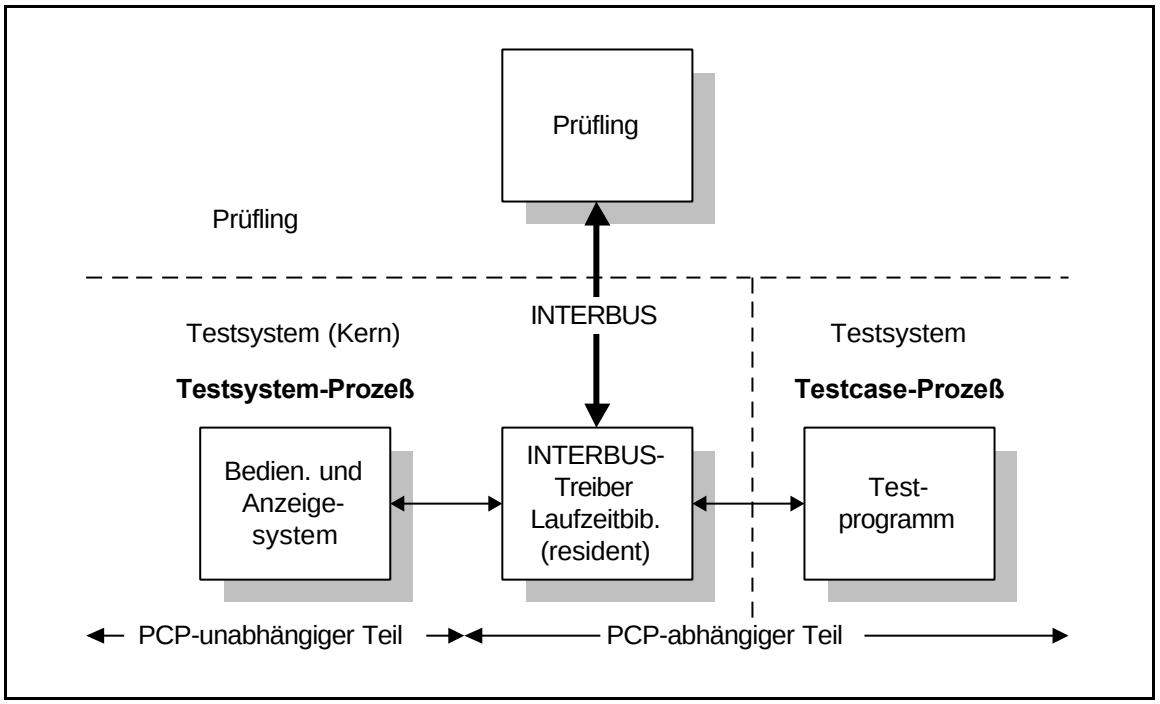

Figure 9: Main procedural framework for test cases

The test system consists of a PCP-independent operating and display system, a PCP-dependent part, essentially comprising the INTERBUS driver, and the test program, which tests the test object for conformity with a particular specification.

6.2.1 Test System Process/Test Case Process

In principle, it would be possible to integrate all the test cases in a single test program and process the individual cases one after the other by means of nested 'switch' instructions. However, there are a few major disadvantages to this approach:

– Memory problems, because a limited amount of memory is available in the test system.

– Long compiling times during program development.

– The need to recompile the test program in the event of changes to the operating and display system.

In the INTERBUS test system, the individual test cases were implemented as independent test case program units to avoid these disadvantages.

Two processes are alternately active in the test system during the test:

– the test system process

– the test case process

The test system process, which contains the operating and display system and the INTERBUS driver, is resident in the test system memory. The test case process is loaded in the memory and started by the test system process, according to the instructions entered by the user. Both processes communicate with one another by calling each other sequentially while transferring parameters. At the end of the test, the test case process terminates automatically.

6.2.2 Interfaces

To further clarify how a test case functions within the test system, Figure 9 is explained in more detail by Figures 10 and 11.

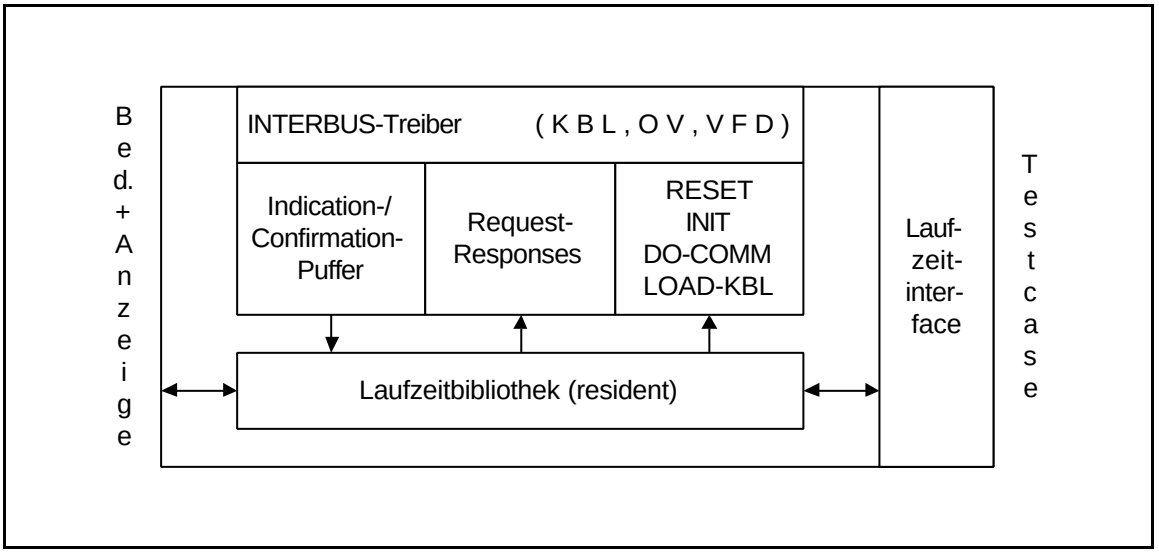

Figure 10: PCP-dependent part of the resident test system

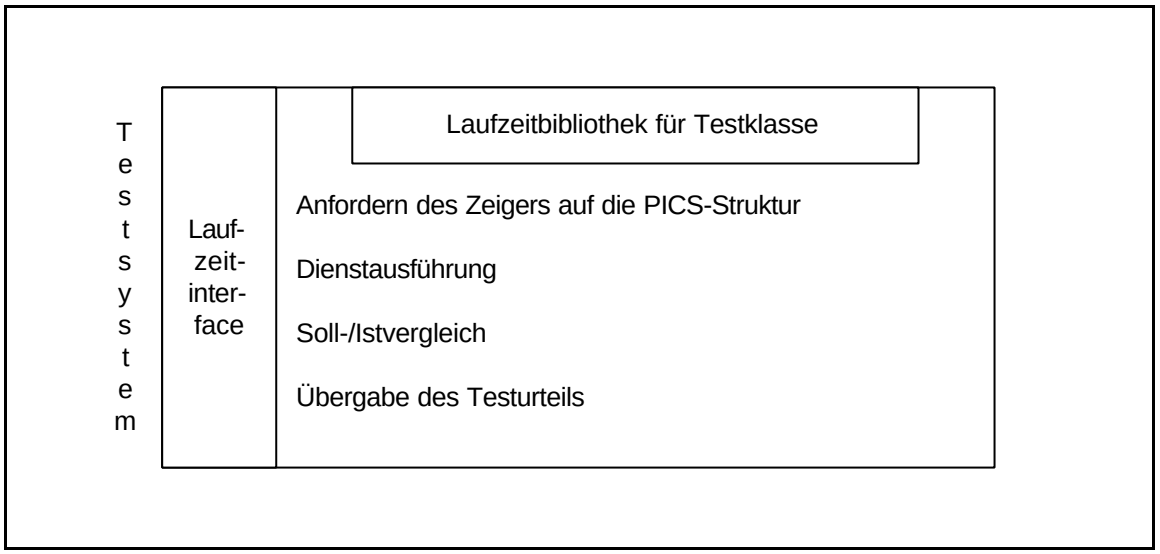

Figure 11: Test case

In the simplest case, a test case is processed as follows:

– The pointer requests the ‘pics’ structure via the run-time interface.

– In the test case, evaluation of the ‘pics’ structure is used to decide whether and with which parameters the test is to be carried out on the test object. If it is not possible to carry out the test, the test result ‘Passed: Full restriction’ is transferred to the test system process and the test case process is ended.

– The request is issued with the prepared service parameters.

– A wait loop is initiated in the test system process for the anticipated confirmation.

– The received confirmation is evaluated using routines from the run-time library in the test case process.

– The test result is determined.

– This result is transferred to the test system process.

6.2.3 Logging

A test case procedure is logged in three ways:

– Logging of the calls to the ALI interface in the test system (1).

– Logging of the calls between the test case and run-time interface (2).

– Logging of the test case procedure implemented by the creator of the test case by means of so-called test messages. The test messages may also belong to various classes: log (3) by means of the call ‘print_to_log’ and log (4) by means of the call ‘print_to_prof_log’.

By means of the ‘loggings’, which can be adjusted in the operating and display system, these logs are used for recording and subsequent evaluation by the user, as follows:

– TEST-LOG: 1+2+3,
– PMS-LOG: 1+3
– and PROF-LOG: 4

Each entry in the log is given a time stamp.

6.2.4 Error Messages

If an error is detected, a test case is generally interrupted immediately, and an error message is issued (this is usually the case in test classes SVN and CLN). Generally speaking, an error message has three components: a reference to the location of the error, e.g. the faulty service parameter in the form of the name, the setpoint (with the prefix ‘E: ’ for expected) and the actual value (with the prefix ‘R: ’ for received). For example:

Failed: locally_generated: E: ff R: 0

In this case, an error was detected by the remote device and connection to the latter was aborted using the abort service, instead of the local device. Interpretation of the error messages at INTERBUS level requires in-depth knowledge of the PCP specification with all its services and parameters.

6.3 Test Results

Each test case tests a well-defined function. It may contain several sub-tests.

In the SVN and CLN test classes, sub-tests may consist of further test sections (these usually contain data pattern tests). Here, all tests are carried out with all the objects specified in the PICS/PIXIT file, even the null objects which are not explicitly implemented. It is true that more computing time is involved, as a function can be tested several times on objects with similar properties, but in certain circumstances this system can detect programming errors. In the case of non-existent objects (the index is not within an OD dictionary range), each test is only carried out once. As a rule, data pattern tests are only carried out once.

The decision as to which tests are to be carried out is taken in each test case relating to the running time by evaluating the internal data structure ‘pics’. For each test case, immediately after processing, a test result is reached in the test case process and transferred to the test system process with any error messages. There are three basic test results:

– passed
– failed
– inconclusive

Group 0 of the test cases for the server and client test tests:

– The implementation of CRL and OD for "unattractiveness" e.g., empty OD areas containing only null objects.
– The executability of the services in the case of the specified or remote-loadable CRL implementation and the specified OD implementation, where all objects are to be accessible for the services.

The object of the tests in this group is the detection of possible shortcomings in implementation. Since these tests do not check for malfunctions in the sense of the PCP specification, warnings are only issued in conjunction with the test result "inconclusive".

Group 0 of the test cases in the profile test tests the PICS/PIXIT file (particularly the profile-specific part) for completeness and consistency with the profile.

6.3.1 Test Result "Passed"

The test result "passed" means that the tests in the test case were unable to detect any errors. This should not be confused with the statement "the test object passed all the tests stipulated in the test case without errors". A few examples are given below:

– Test cases may contain sub-tests which mutually exclude one another in the case of a test object.

– Tests do not need to be carried out for a test object if the functions to be tested are optional and are not specified for the test object (e.g. optional services).

– Tests cannot be carried out if there are no corresponding objects in the object dictionary of the test object which enable the test to be carried out.

Nevertheless, the test will always receive the result "passed", as the function to be tested is not operated during the actual use of the device.

However, for further grading of a successful test, there are additional details which provide information about the degree of coverage of the implemented or useable function in the test object:

– no restrictions
– some restrictions (PICS: number)
– full restriction (PICS)

The test result "Passed: No restrictions" means that all the tests implemented in the test case were carried out and no errors were detected in the test object.
The test result "Passed: Some restrictions" means that not all the tests implemented in the test case were carried out, but the tests which were carried out did not detect any errors in the test object. The number of tests which were not carried out is specified.

The test result "Passed: Full restriction" means that none of the implemented tests could be carried out in the test case.

6.3.2 Test Result "Failed"

The test result "Failed" means that a test in the test case has detected an error. The test case will be interrupted immediately after detection of the first error. However, the elimination of an error does not mean that the test case will then run smoothly, since all subsequent tests can only be activated once the error has been eliminated.

6.3.3 Test Result "Inconclusive"

The test result "Inconclusive" means that, ultimately, it was not possible to reach an automatic conclusion in terms of "Passed" or "Failed". In the case of an "Inconclusive" result, it is necessary for an examiner to judge this test result on the basis of the PICS file and the test log. The test result "Inconclusive" is also always given if permissible access operations to objects of the object dictionary are prohibited by the options NO_R, NO_W, NO_RW and NO_SH. However, there are further details which make the decision slightly easier for the person carrying out the test:

– some restrictions (USER: number)
– some restrictions (PICS: number, USER: number)
– and full restriction (PICS, USER)

The test result "Inconclusive: Some restrictions (USER)" means that all the tests implemented in the test case were carried out and no errors were detected in the test object, but that these tests were not carried out for all the objects defined in the PICS file to which these tests could have been applicable. The number of objects which the creator of the PICS file (USER) excluded from the test is specified.

The test result "Inconclusive: Some restrictions (PICS, USER)" means that not all the tests implemented in the test case were carried out, but the tests which were carried out did not detect any errors in the test object. Here too, these tests were not carried out for all the objects defined in the PICS file to which these tests could have been applicable. However, in contrast to the previous result, not all the implemented tests were carried out. The examiner should therefore check whether the degree of coverage could have been increased if objects had not been excluded from the test. The number of tests which were not carried out and the number of objects which the creator of the PICS file (USER) excluded from the test is specified.

The test result "Inconclusive: Full restriction (PICS, USER)" means that none of the implemented tests could be carried out in the test case. The examiner should therefore check if it would have been possible to carry out the tests if no objects had been excluded from the test. The number of tests which were not carried out and the number of objects which the creator of the PICS file (USER) excluded from the test is specified.

In all cases, a decisive factor enabling the examiner to change the test result from "Inconclusive" to "Passed" is the indication that the degree of coverage was not influenced by the exclusion of objects from the test.

Other "Inconclusive" test results indicate that a test could not be carried out because of certain other conditions. For example:

– an event which cannot be influenced by the test system did not occur, e.g. an information report indication was not received.

The examiner must evaluate the message transmitted with the "Inconclusive" test result on an individual basis and reach his decision.

6.4 Certification Procedure

The certification procedure is carried out using the certification schedule of the respective test class. The test cases in the certification schedule are carried out in the order in which they are included and a test result is formed for each test case. The individual test results are saved as test reports in the ‘result’ file. When the test schedules have been processed, each test case which has been carried out is allocated one of the results ‘Passed’, ‘Failed’ or ‘Inconclusive’. The test case results in the ‘result’ file are arranged alphanumerically. Test cases which belong to the respective test class but which are not included in the certification schedule will not be given a test result. They are not relevant to the issuing of the certificate. Such test cases are generally provided for later versions and may also be used for

the development test. As well as the test results, the test report contains statistical data relating to the test procedure and the description of the test object in the PICS/PIXIT file on which the test was based.

Certificates will be issued by the respective certification body upon submission of the test report (with the associated log data) from the test laboratory. A tested device will receive the certificate for a particular test class if all the test cases of the certification schedule for this test class received the test result ‘passed’. Generally speaking, the certificate will not be issued if even one of the test cases ‘failed’. In the case of an ‘inconclusive’ test result, it is not possible to reach a simple verdict. In this instance, the certification body must decide on a case-by-case basis, where applicable by means of arbitration.

7 Appendix A: Test Case List for PCP 2,0

When selecting the certification schedule, the following test object properties must be taken into account:

1. Test object without remote management
2. Test object with remote management,
   load_kbl_rem service not supported
3. Test object with remote management,
   load_kbl_rem service supported

For a conformance test of PCP Version 2.0 for test objects without remote management, the following 241 test cases must be carried out:

1.	CTPMS	31.	SVN13030	61.	SVN30400	91.	SVN44200
2.	SVN00010	32.	SVN13040	62.	SVN30420	92.	SVN40000
3.	SVN00020	33.	SVN13120	63.	SVN30430	93.	SVN40020
4.	SVN01000	34.	SVN13140	64.	SVN30520	94.	SVN40030
5.	SVN01010	35.	SVN13220	65.	SVN30530	95.	SVN40040
6.	SVN01100	36.	SVN13230	66.	SVN30540	96.	SVN40100
7.	SVN01110	37.	SVN13240	67.	SVN30550	97.	SVN40110
8.	SVN02000	38.	SVN14000	68.	SVN30560	98.	SVN40120
9.	SVN02010	39.	SVN14010	69.	SVN30570	99.	SVN40130
10.	SVN02020	40.	SVN14020	70.	SVN30580	100.	SVN40140
11.	SVN02030	41.	SVN14030	71.	SVN30590	101.	SVN40160
12.	SVN03000	42.	SVN20000	72.	SVN30600	102.	SVN40170
13.	SVN03010	43.	SVN21000	73.	SVN31010	103.	SVN40180
14.	SVN03020	44.	SVN30000	74.	SVN31030	104.	SVN40200
15.	SVN03030	45.	SVN30020	75.	SVN31050	105.	SVN41000
16.	SVN03040	46.	SVN30040	76.	SVN31070	106.	SVN41020
17.	SVN06000	47.	SVN30060	77.	SVN31080	107.	SVN41030
18.	SVN07000	48.	SVN30070	78.	SVN31090	108.	SVN41040
19.	SVN10000	49.	SVN30080	79.	SVN31110	109.	SVN41100
20.	SVN10020	50.	SVN30100	80.	SVN31130	110.	SVN41110
21.	SVN10030	51.	SVN30120	81.	SVN31150	111.	SVN41120
22.	SVN10050	52.	SVN30140	82.	SVN31170	112.	SVN41130
23.	SVN11000	53.	SVN30160	83.	SVN31190	113.	SVN41140
24.	SVN11010	54.	SVN30180	84.	SVN31270	114.	SVN41160
25.	SVN11040	55.	SVN30260	85.	SVN31280	115.	SVN41170
26.	SVN12080	56.	SVN30270	86.	SVN31290	116.	SVN41180
27.	SVN12100	57.	SVN30280	87.	SVN31310	117.	SVN41200
28.	SVN13000	58.	SVN30300	88.	SVN31330	118.	SVN42000
29.	SVN13010	59.	SVN30320	89.	SVN31350	119.	SVN42020
30.	SVN13020	60.	SVN30340	90.	SVN31410	120.	SVN42030

121.	SVN42040	152.	SVN50100	183.	SVN51060	214.	SVN51630
122.	SVN42100	153.	SVN50110	184.	SVN51100	215.	SVN51640
123.	SVN42110	154.	SVN50120	185.	SVN51110	216.	SVN51650
124.	SVN42120	155.	SVN50130	186.	SVN51120	217.	SVN51700
125.	SVN42130	156.	SVN50140	187.	SVN51130	218.	SVN51710
126.	SVN42140	157.	SVN50150	188.	SVN51140	219.	SVN51730
127.	SVN42160	158.	SVN50200	189.	SVN51150	220.	SVN51800
128.	SVN42170	159.	SVN50210	190.	SVN51160	221.	SVN51810
129.	SVN42180	160.	SVN50220	191.	SVN51200	222.	SVN51820
130.	SVN42200	161.	SVN50230	192.	SVN51210	223.	SVN51830
131.	SVN43000	162.	SVN50240	193.	SVN51220	224.	SVN51860
132.	SVN43020	163.	SVN50250	194.	SVN51230	225.	SVN51870
133.	SVN43030	164.	SVN50300	195.	SVN51240	226.	SVN51880
134.	SVN43040	165.	SVN50320	196.	SVN51250	227.	SVN52000
135.	SVN43100	166.	SVN50330	197.	SVN51260	228.	SVN52020
136.	SVN43110	167.	SVN50340	198.	SVN51300	229.	SVN52030
137.	SVN43120	168.	SVN50400	199.	SVN51320	230.	SVN52040
138.	SVN43130	169.	SVN50500	200.	SVN51330	231.	SVN52400
139.	SVN43140	170.	SVN50600	201.	SVN51340	232.	SVN53000
140.	SVN43160	171.	SVN50700	202.	SVN51360	233.	SVN53020
141.	SVN43170	172.	SVN50800	203.	SVN51370	234.	SVN53030
142.	SVN43180	173.	SVN50810	204.	SVN51400	235.	SVN53040
143.	SVN43200	174.	SVN50820	205.	SVN51410	236.	SVN53060
144.	SVN44000	175.	SVN50830	206.	SVN51430	237.	SVN53400
145.	SVN44010	176.	SVN50860	207.	SVN51500	238.	SVN53440
146.	SVN44050	177.	SVN50870	208.	SVN51510	239.	SVN53450
147.	SVN44100	178.	SVN50880	209.	SVN51530	240.	SVN54000
148.	SVN50000	179.	SVN51000	210.	SVN51540	241.	SVN62150
149.	SVN50020	180.	SVN51020	211.	SVN51550		
150.	SVN50030	181.	SVN51030	212.	SVN51600		
151.	SVN50040	182.	SVN51040	213.	SVN51610		

For a conformance test of PCP Version 2.0 for test objects with remote management and in which the load_kbl_rem service is not supported, the following 293 test cases must be carried out:

1.	CTPMSM	11.	SVN02030	21.	SVN10030	31.	SVN13030
2.	SVN00010	12.	SVN03000	22.	SVN10050	32.	SVN13040
3.	SVN00020	13.	SVN03010	23.	SVN11000	33.	SVN13120
4.	SVN01000	14.	SVN03020	24.	SVN11010	34.	SVN13140
5.	SVN01010	15.	SVN03030	25.	SVN11040	35.	SVN13220
6.	SVN01100	16.	SVN03040	26.	SVN12080	36.	SVN13230
7.	SVN01110	17.	SVN06000	27.	SVN12100	37.	SVN13240
8.	SVN02000	18.	SVN07000	28.	SVN13000	38.	SVN14000
9.	SVN02010	19.	SVN10000	29.	SVN13010	39.	SVN14010
10.	SVN02020	20.	SVN10020	30.	SVN13020	40.	SVN14020

41.	SVN14030	85.	SVN31280	129.	SVN42180	173.	SVN50810
42.	SVN20000	86.	SVN31290	130.	SVN42200	174.	SVN50820
43.	SVN21000	87.	SVN31310	131.	SVN43000	175.	SVN50830
44.	SVN30000	88.	SVN31330	132.	SVN43020	176.	SVN50860
45.	SVN30020	89.	SVN31350	133.	SVN43030	177.	SVN50870
46.	SVN30040	90.	SVN31410	134.	SVN43040	178.	SVN50880
47.	SVN30060	91.	SVN44200	135.	SVN43100	179.	SVN51000
48.	SVN30070	92.	SVN40000	136.	SVN43110	180.	SVN51020
49.	SVN30080	93.	SVN40020	137.	SVN43120	181.	SVN51030
50.	SVN30100	94.	SVN40030	138.	SVN43130	182.	SVN51040
51.	SVN30120	95.	SVN40040	139.	SVN43140	183.	SVN51060
52.	SVN30140	96.	SVN40100	140.	SVN43160	184.	SVN51100
53.	SVN30160	97.	SVN40110	141.	SVN43170	185.	SVN51110
54.	SVN30180	98.	SVN40120	142.	SVN43180	186.	SVN51120
55.	SVN30260	99.	SVN40130	143.	SVN43200	187.	SVN51130
56.	SVN30270	100.	SVN40140	144.	SVN44000	188.	SVN51140
57.	SVN30280	101.	SVN40160	145.	SVN44010	189.	SVN51150
58.	SVN30300	102.	SVN40170	146.	SVN44050	190.	SVN51160
59.	SVN30320	103.	SVN40180	147.	SVN44100	191.	SVN51200
60.	SVN30340	104.	SVN40200	148.	SVN50000	192.	SVN51210
61.	SVN30400	105.	SVN41000	149.	SVN50020	193.	SVN51220
62.	SVN30420	106.	SVN41020	150.	SVN50030	194.	SVN51230
63.	SVN30430	107.	SVN41030	151.	SVN50040	195.	SVN51240
64.	SVN30520	108.	SVN41040	152.	SVN50100	196.	SVN51250
65.	SVN30530	109.	SVN41100	153.	SVN50110	197.	SVN51260
66.	SVN30540	110.	SVN41110	154.	SVN50120	198.	SVN51300
67.	SVN30550	111.	SVN41120	155.	SVN50130	199.	SVN51320
68.	SVN30560	112.	SVN41130	156.	SVN50140	200.	SVN51330
69.	SVN30570	113.	SVN41140	157.	SVN50150	201.	SVN51340
70.	SVN30580	114.	SVN41160	158.	SVN50200	202.	SVN51360
71.	SVN30590	115.	SVN41170	159.	SVN50210	203.	SVN51370
72.	SVN30600	116.	SVN41180	160.	SVN50220	204.	SVN51400
73.	SVN31010	117.	SVN41200	161.	SVN50230	205.	SVN51410
74.	SVN31030	118.	SVN42000	162.	SVN50240	206.	SVN51430
75.	SVN31050	119.	SVN42020	163.	SVN50250	207.	SVN51500
76.	SVN31070	120.	SVN42030	164.	SVN50300	208.	SVN51510
77.	SVN31080	121.	SVN42040	165.	SVN50320	209.	SVN51530
78.	SVN31090	122.	SVN42100	166.	SVN50330	210.	SVN51540
79.	SVN31110	123.	SVN42110	167.	SVN50340	211.	SVN51550
80.	SVN31130	124.	SVN42120	168.	SVN50400	212.	SVN51600
81.	SVN31150	125.	SVN42130	169.	SVN50500	213.	SVN51610
82.	SVN31170	126.	SVN42140	170.	SVN50600	214.	SVN51630
83.	SVN31190	127.	SVN42160	171.	SVN50700	215.	SVN51640
84.	SVN31270	128.	SVN42170	172.	SVN50800	216.	SVN51650

217.	SVN51700	237.	SVN53400	257.	SVN71110	277.	SVN72240
218.	SVN51710	238.	SVN53440	258.	SVN71120	278.	SVN72250
219.	SVN51730	239.	SVN53450	259.	SVN70020	279.	SVN72260
220.	SVN51800	240.	SVN54000	260.	SVN72000	280.	SVN72270
221.	SVN51810	241.	SVN62150	261.	SVN72010	281.	SVN72280
222.	SVN51820	242.	SVN60000	262.	SVN72030	282.	SVN72290
223.	SVN51830	243.	SVN60020	263.	SVN72100	283.	SVN72310
224.	SVN51860	244.	SVN60050	264.	SVN72110	284.	SVN72320
225.	SVN51870	245.	SVN61000	265.	SVN72120	285.	SVN80000
226.	SVN51880	246.	SVN61040	266.	SVN72020	286.	SVN80010
227.	SVN52000	247.	SVN62080	267.	SVN72130	287.	SVN80020
228.	SVN52020	248.	SVN62100	268.	SVN72140	288.	SVN80030
229.	SVN52030	249.	SVN64000	269.	SVN72150	289.	SVN80070
230.	SVN52040	250.	SVN64010	270.	SVN72160	290.	SVN80080
231.	SVN52400	251.	SVN64020	271.	SVN72170	291.	SVN80100
232.	SVN53000	252.	SVN64030	272.	SVN72180	292.	SVN80110
233.	SVN53020	253.	SVN70000	273.	SVN72190	293.	SVN80300
234.	SVN53030	254.	SVN70010	274.	SVN72200		
235.	SVN53040	255.	SVN71000	275.	SVN72210		
236.	SVN53060	256.	SVN71100	276.	SVN72220		

For a conformance test of PCP Version 2.0 for test objects with remote management and in which the load_kbl_rem service is supported, the following 323 test cases must be carried out:

1.	CTPMSML	42.	SVN72120	83.	SVN13020	124.	SVN30590
2.	SVN00010	43.	SVN72020	84.	SVN13030	125.	SVN30600
3.	SVN00020	44.	SVN72130	85.	SVN13040	126.	SVN31010
4.	SVN01000	45.	SVN72140	86.	SVN13120	127.	SVN31030
5.	SVN01010	46.	SVN72150	87.	SVN13140	128.	SVN31050
6.	SVN01100	47.	SVN72160	88.	SVN13220	129.	SVN31070
7.	SVN01110	48.	SVN72170	89.	SVN13230	130.	SVN31080
8.	SVN02000	49.	SVN72180	90.	SVN13240	131.	SVN31090
9.	SVN02010	50.	SVN72190	91.	SVN14000	132.	SVN31110
10.	SVN02020	51.	SVN72200	92.	SVN14010	133.	SVN31130
11.	SVN02030	52.	SVN72210	93.	SVN14020	134.	SVN31150
12.	SVN03000	53.	SVN72220	94.	SVN14030	135.	SVN31170
13.	SVN03010	54.	SVN72240	95.	SVN20000	136.	SVN31190
14.	SVN03020	55.	SVN72250	96.	SVN21000	137.	SVN31270
15.	SVN03030	56.	SVN72260	97.	SVN30000	138.	SVN31280
16.	SVN03040	57.	SVN72270	98.	SVN30020	139.	SVN31290
17.	SVN06000	58.	SVN72280	99.	SVN30040	140.	SVN31310
18.	SVN07000	59.	SVN72290	100.	SVN30060	141.	SVN31330
19.	SVN60000	60.	SVN72310	101.	SVN30070	142.	SVN31350
20.	SVN60020	61.	SVN72320	102.	SVN30080	143.	SVN31410
21.	SVN60050	62.	SVN80000	103.	SVN30100	144.	SVN44200
22.	SVN61000	63.	SVN80010	104.	SVN30120	145.	SVN40000
23.	SVN61040	64.	SVN80020	105.	SVN30140	146.	SVN40020
24.	SVN62080	65.	SVN80030	106.	SVN30160	147.	SVN40030
25.	SVN62100	66.	SVN80070	107.	SVN30180	148.	SVN40040
26.	SVN64000	67.	SVN80080	108.	SVN30260	149.	SVN40100
27.	SVN64010	68.	SVN80100	109.	SVN30270	150.	SVN40110
28.	SVN64020	69.	SVN80110	110.	SVN30280	151.	SVN40120
29.	SVN64030	70.	SVN80300	111.	SVN30300	152.	SVN40130
30.	SVN70000	71.	KBL0	112.	SVN30320	153.	SVN40140
31.	SVN70010	72.	SVN10000	113.	SVN30340	154.	SVN40160
32.	SVN71000	73.	SVN10020	114.	SVN30400	155.	SVN40170
33.	SVN71100	74.	SVN10030	115.	SVN30420	156.	SVN40180
34.	SVN71110	75.	SVN10050	116.	SVN30430	157.	SVN40200
35.	SVN71120	76.	SVN11000	117.	SVN30520	158.	SVN41000
36.	SVN70020	77.	SVN11010	118.	SVN30530	159.	SVN41020
37.	SVN72000	78.	SVN11040	119.	SVN30540	160.	SVN41030
38.	SVN72010	79.	SVN12080	120.	SVN30550	161.	SVN41040
39.	SVN72030	80.	SVN12100	121.	SVN30560	162.	SVN41100
40.	SVN72100	81.	SVN13000	122.	SVN30570	163.	SVN41110
41.	SVN72110	82.	SVN13010	123.	SVN30580	164.	SVN41120

165.	SVN41130	210.	SVN50150	255.	SVN51360	300.	SVN13011
166.	SVN41140	211.	SVN50200	256.	SVN51370	301.	SVN13021
167.	SVN41160	212.	SVN50210	257.	SVN51400	302.	SVN13031
168.	SVN41170	213.	SVN50220	258.	SVN51410	303.	SVN13041
169.	SVN41180	214.	SVN50230	259.	SVN51430	304.	SVN13121
170.	SVN41200	215.	SVN50240	260.	SVN51500	305.	SVN13141
171.	SVN42000	216.	SVN50250	261.	SVN51510	306.	SVN13221
172.	SVN42020	217.	SVN50300	262.	SVN51530	307.	SVN13231
173.	SVN42030	218.	SVN50320	263.	SVN51540	308.	SVN13241
174.	SVN42040	219.	SVN50330	264.	SVN51550	309.	SVN14011
175.	SVN42100	220.	SVN50340	265.	SVN51600	310.	SVN20001
176.	SVN42110	221.	SVN50400	266.	SVN51610	311.	SVN21001
177.	SVN42120	222.	SVN50500	267.	SVN51630	312.	SVN30001
178.	SVN42130	223.	SVN50600	268.	SVN51640	313.	SVN30401
179.	SVN42140	224.	SVN50700	269.	SVN51650	314.	SVN31011
180.	SVN42160	225.	SVN50800	270.	SVN51700	315.	SVN31411
181.	SVN42170	226.	SVN50810	271.	SVN51710	316.	SVN50151
182.	SVN42180	227.	SVN50820	272.	SVN51730	317.	SVN50251
183.	SVN42200	228.	SVN50830	273.	SVN51800	318.	SVN51151
184.	SVN43000	229.	SVN50860	274.	SVN51810	319.	SVN51251
185.	SVN43020	230.	SVN50870	275.	SVN51820	320.	KBL2
186.	SVN43030	231.	SVN50880	276.	SVN51830	321.	SVN10002
187.	SVN43040	232.	SVN51000	277.	SVN51860	322.	SVN10032
188.	SVN43100	233.	SVN51020	278.	SVN51870	323.	SVN11002
189.	SVN43110	234.	SVN51030	279.	SVN51880		
190.	SVN43120	235.	SVN51040	280.	SVN52000		
191.	SVN43130	236.	SVN51060	281.	SVN52020		
192.	SVN43140	237.	SVN51100	282.	SVN52030		
193.	SVN43160	238.	SVN51110	283.	SVN52040		
194.	SVN43170	239.	SVN51120	284.	SVN52400		
195.	SVN43180	240.	SVN51130	285.	SVN53000		
196.	SVN43200	241.	SVN51140	286.	SVN53020		
197.	SVN44000	242.	SVN51150	287.	SVN53030		
198.	SVN44010	243.	SVN51160	288.	SVN53040		
199.	SVN44050	244.	SVN51200	289.	SVN53060		
200.	SVN44100	245.	SVN51210	290.	SVN53400		
201.	SVN50000	246.	SVN51220	291.	SVN53440		
202.	SVN50020	247.	SVN51230	292.	SVN53450		
203.	SVN50030	248.	SVN51240	293.	SVN54000		
204.	SVN50040	249.	SVN51250	294.	SVN62150		
205.	SVN50100	250.	SVN51260	295.	KBL1		
206.	SVN50110	251.	SVN51300	296.	SVN10001		
207.	SVN50120	252.	SVN51320	297.	SVN10051		
208.	SVN50130	253.	SVN51330	298.	SVN11041		
209.	SVN50140	254.	SVN51340	299.	SVN13001		

8 Notes