Title: **Requirements IPA Wanda Lao**

Subject: **Requirements , Specification**

This document is an extract of the document TsNet V2 Requirements. It contains the requirements which are relevant for the IPA.

Key Words: TsNet V2

Document Storage:	Local
Document Category:	ProjectRecord
Revision:	2
Revision Date:	2017-03-04
Document Status:	Final - without Approval
Author:	Michael Speckien, 5556
Department:	IC BT CPS R&D ZG CS SAP
Responsible:	Michael.Speckien@siemens.com
Company:	Siemens Schweiz AG, Building Technologies Division Control Products & Systems
Based on Template:	Workbook_Small; 4; 2014-11-05; Donat Hutter, 3531

Revision History

Rev	Date	Author	Remarks
2	04-Mrz-2017	Michael Speckien, 5556	Status = **Final - without Approval** - Ready for IPA
1	03-Mrz-2017	Urs Heimann	Status = **Working**

Table of Contents

1. Introduction

1.1 Purpose of the document

This document serves as requirement based specification for the IPA. It is an extract from the overall document TsNet V2 Requirements [1], which is still in status “working”

1.2 Scope, Field of application

The task for this IPA is a dedicated part of the development TsNet V2.

1.3 Document References

[1] TsNet V2 Requirements, Michael Speckien 2016

[2] PAL-GL-0012_DE_CodierungsrichtlinieVisualBasic.doc [Link]

1.4 Definitions, Acronyms and Abbreviations, Conventions

1.4.1 Glossary

Term	Description

1.4.2 Conventions in this Document

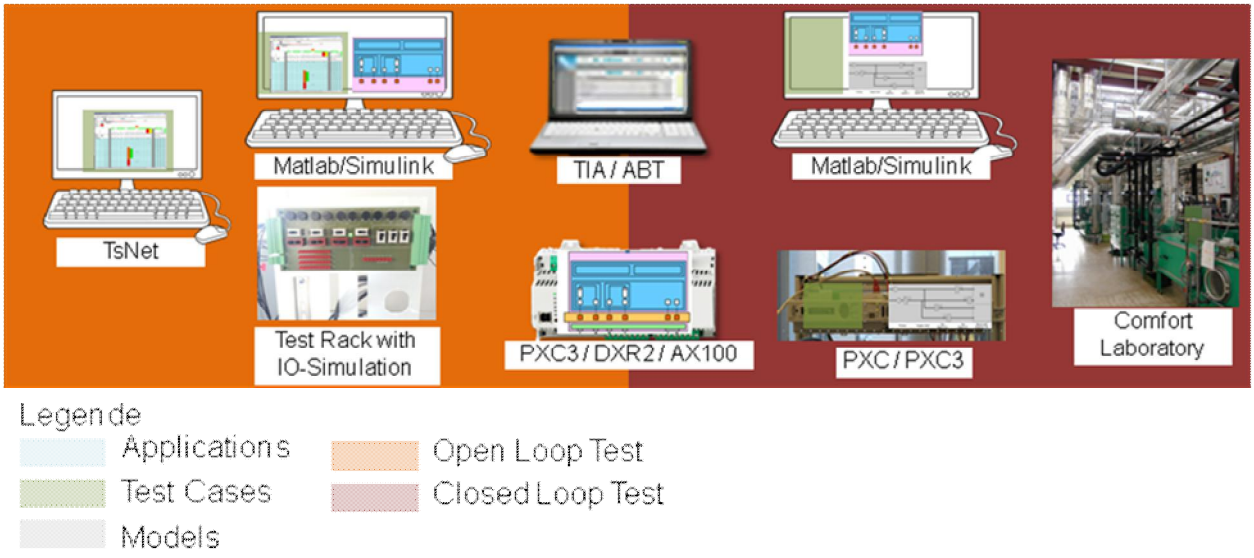
Issues which are interesting for understanding the complete TsNet V2 project, but which are not relevant for the IPA are labeled with “Not relevant for IPA”

1.5 Open Issues in this Document Version

No open issues, this document is valid for the IPA.

2. Actual situation

2.1 Overview TsNet



TsNet is a pc running software package consisting of a test specification part (Excel based template) and a runtime part (test operation and BACnet stack)
TsNet allows executing application tests with hardware-in-the-loop by triggering variables via BACnet, reading the controller's reaction via BACnet and comparing it with the expected results.
TsNet in application testing is usually combined with a test rack, allowing additional triggering via physical inputs and easy monitoring of physical output signals.
Actually TsNet is mainly used for open-loop-tests, however it can be combined with a room simulation on a controller.

3. Requirements

The actual version of TsNet should get some improvements to simplify the dayly use in application testing.
This chapter is an extract of [1]. It shows only requirements which are relevant for the IPA.

TsNetV2-0020	**Functional Test Specification (independent of the test environment)**
DESCR:	The test specification actually is partly depending on the ABT engineering data, especially datapoint names. So when changing ABT engineering data, it is required to rework the test specification. This increases the work required for regression tests. → It should be possible to change datapoint names and controller names without reworking the test specification
PRIO:	0
REF:	Chapter 4.5, 5.1
TEST:	Covered by the specification in 5.1. No specific test required

TsNetV2-0050	**Workflow orientation**
DESCR:	TsNet specification template does not support a defined workflow. So it is the user's task to maintain consistent data. → TsNet sheets shall include a change control and a status information. It shall indicate possibly inconsistent data and show the user how to get the data consistent.
PRIO:	0
REF:	Chapter 4.3, 5.1.2
TEST:	Follow the described workflow in 5.1.2 and verify the status of the sheet and the GUI according to the description of the functions "Modify", "Check List", "Check Devices". Verify the data in the fileds "Status", "Info" and "Status Device"

TsNetV2-0060	**Programming guideline**
DESCR:	The template has been developed over a long period with different developers resulting in different programming styles and different user interfaces, for example limitations in copy/paste. → The template should be redesigned following programming guidelines.
PRIO:	0
REF:	Chapter 4.2, 4.4
TEST:	Non-functional requirement, not to be tested

4. Design proposal for TsNet Test Specification Template

Fulfilling the requirements, a complete redesign of the TsNet specification template is required.

4.1 Location and file structure

The tool TsNet runtime is delivered with an installation routine.
The test specification template can be copied to any location on the disk or a server and shall be renamed to a project specific name like Test_AF_CenOpMod11.xlsm.
The complete data for running the tests is stored in the different sheets of the excel file.
*: Worksheets marked with * are not relevant for the IPA.

Structure of the Excel file:

Excel file	Worksheets	Description
Excel_file.xlsm	Config*	Basic configuration of the test
	Overview*	Main sheet for navigating, generating, executing and documenting tests
	Workflow*	Documentation of the TsNet V2 workflow
	Spec-Devices	List of the BACnet devices used in the test
	Spec-Objects	**List of all BACnet objects used in the test This sheet and its functions shall be created in the IPA**
	For each test step	
	Step_<name>_<src>*	Specification of the test step source, can be in the form of a text (_TXT), or a vertical table (_VT)
	Step_<name>_VT*	Generated from <src> if <src> is TXT format. Required before executing the test.
	Step_<name>_Trend*	Optional generated from VT: Trend view of the specification for documentation
	Step_<name>_Script*	Generated from VT. Required before executing the test Data for TsNet Runtime
	Step_<name>_Result*	Generated after test execution from TsNet Runtime
	Imported data from external tools	
	EDE	EDE list from engineering tool
	Data for external tools	
	QCdata*	Export data for QC (HP ALM)
	POOSdata*	Specification file for data logging tool POOS
	Templates used for creating empty sheets for test steps	
	Template Step_TXT*	Empty template for test spec as text
	Template Step_VT*	Empty template for test spec as vertical table
	Template Step_Trend*	Empty template for trend view
	Template Step_Script*	Template for test script
	Template Step_Result*	Template for result
	Templates used for generating external data	
	Template QC*	Template for QC
	Template POOS*	Template for POOS
	Internal sheets	
	Enum	BACnet enumerations for objects, properties etc
	Help	Sheet with all help references
	Choices*	Non-BACnet enumerations for the user interface

		SDU*		Shortname and description from “All_Translatable_HQ_Texts_ger_20151201.xls”

4.2 User Interface Rules

General improvements to the User Interface

- Follow the standard convention for input fields:
  white = user input
  grey = read-only

QC Testname	011 AF CenOpMod11 - MT
QC Login	speckiem

- Offer navigation features on each sheet
  Navigation to overview sheet and to related sheets

Step No	Step name
1	Init
2	WoBoost
3	Boost

→ link to sheet with Boost

- Each sheet shall be split up into a functional area (grey) and an input area (white)

c:\Programme (x86)\Siemens\TsNet		TsNet Test-PC	192.168.0.233
Alias Controller	**Controllername**	**Comment**	**IP-Addr**
Supervisory	Sprv100	Supervisory functions	192.168.0.100
Central	Cen050	Central functions	192.168.0.50

- Functional area:
  Show information required
  Buttons / links for navigation
  Buttons for calling functions
  Read-only
  Adding cells, columns, lines not possible

- Input area:
  Lists with user data
  Usually horizontal orientation
  Standard Excel function for add lines, filter, sort, fill
  Some lists also allow adding columns

- Some functions allow operation on one or multiple lines or columns.
  Functions for select:
  Selected items are marked with a ✓
  Selecting an item is possible by selecting the line / column with the item to select or by selecting a cell in the line / column of the item to select.
  - Select a line / column and click the ✓ button.
    The selected item is added to the selection.
  - Select a line / column and click the x button.
    The selected item is removed from the selection

- Select multiple lines / columns and click the ✓ button.
  All selected items are added to the selection.
- Select multiple lines / columns and click the x button.
  All selected items are removed from the selection
- Select one or multiple lines / columns and call a function for selected items like “Delete test step”
  If the function called allows multiple selections (like “Run selected test steps”), the previous selection is replaced by all of the selected items.
  If the function allows only 1-n selection like “Insert test step after selected”, the previous selection is replaced by the first of the selected items.

  Proposal:

Select ✓ x	Alias Controller
✓	Supervisory
	Central
	Room

- Help
  Use PDF document with link to a specific page.
  Advantages:
  - use graphics, screenshots etc
  - create help from word document automatically
  - Help function once on each sheet

  For easier maintenance, a Help sheet is used where PDF file and the page for each sheet is defined.

4.3 Workflow orientation (Req 50)

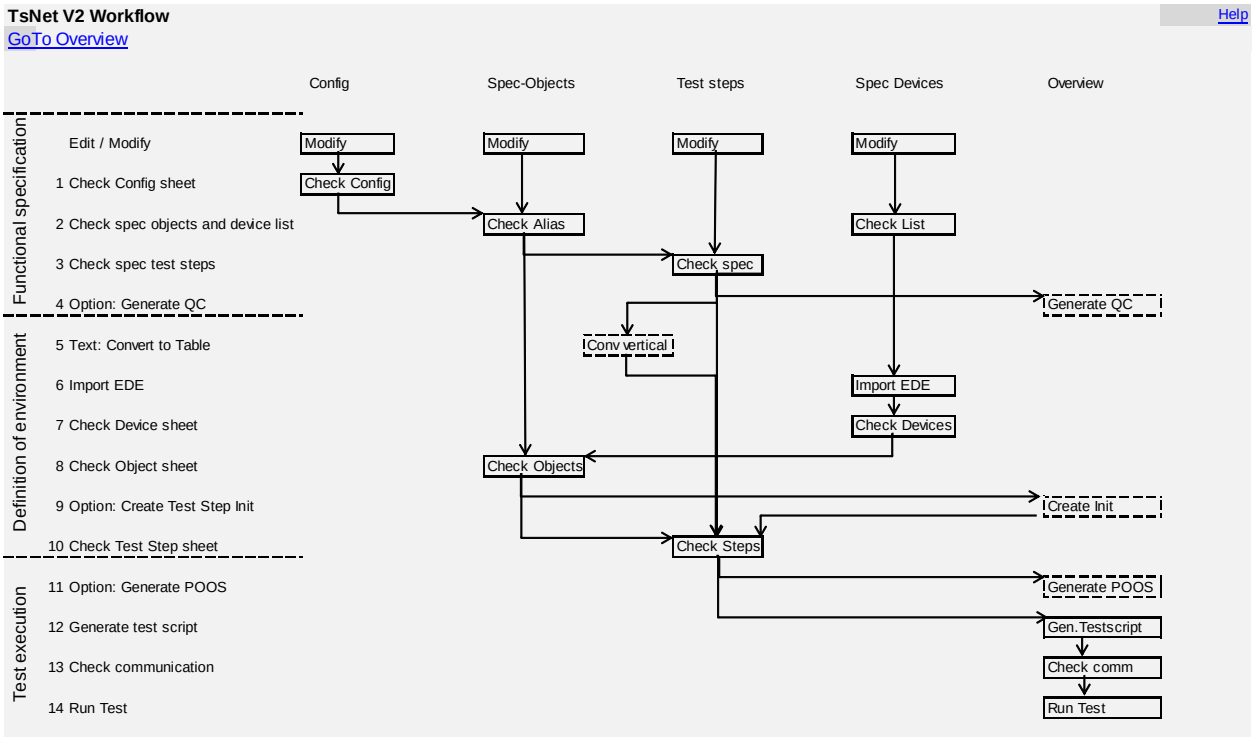
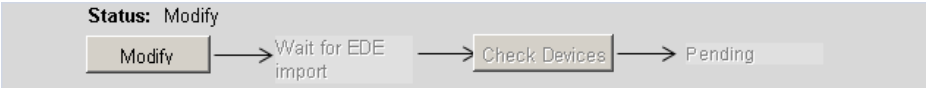


Figure 4–1: Main workflow

The workflow shown is the “straight forward” workflow.
To allow later changes without getting inconsistent, it is required to track changes.

Proposed solution:
Data input into a sheet is followed by a check. After the check has been successfully performed, the sheet is set to the “Checked” state and to read-only. Modification by the user is possible by setting the sheet to the “Modify” state. After a change, the check has to be repeated to get back to the “Checked” state. Each sheet shall show the actual state of the sheet and which steps the user has to execute to follow the workflow including dependencies from other sheets.



!!! The template does not force the user to follow the workflow!!!
It must be possible to ignore the checks and to proceed without having the previous steps done, for example continue running a test immediately after a change in a test step has been made. In this case it is the user’s responsibility to ensure the consistency of the data.

4.4 Programming Guidelines (Req 60)

Please follow the SBT guidelines in [2].

Here are some additional recommendations to follow.

4.4.1 Avoid formulas

A major concern in the actual TsNet version is the use of formulas within an excel sheet. Formulas are difficult to maintain. For the user, formulas within a sheet make it impossible to use standard excel functions like insert line, delete column, copy/paste etc.
Proposed solution:
To verify user’s input or to calculate values, buttons with VBA code are used within the sheet. Check-buttons also support the workflow orientation.

4.4.2 Names for Cells and Ranges

Named ranges for the complete Excel file shall only be used for
- dropdown menus (when the list is on another sheet, names must be used)

All other named ranges shall be local to the worksheet, for example
- definition of writeable areas and read-only areas on a sheet

Restrictions:
Most of the sheets have lists of undefined length, like the list of devices..
No Named ranges must be used for cells or ranges in these areas, as this creates problems when the user copy/paste lines or columns.

For lists, the ranges shall be identified by a named cell and empty lines / columns.
The beginning of a list is identified by a named cell above / left of the list. This should be in the read-only area of the worksheet. The cell name shall be also visible in the cell’s value.
The first empty line / column in a list is considered as the end of the list.
Example: Named Cell: StartList.
The list starts 1 line below StartList and ends with the last non-empty line.
The range is marked yellow

Select	Alias Controller	Device-Name

StartList		
✓	Central	Cen050
✓	Room	Room053
✓	Sprv	Sprv100
	Segment	Segm054
	Router	ROUTER1

4.4.3 Addressing of Cells within a range

Goal:
- Avoid use of too many named ranges
- Give flexibility for inserting lines and columns
- Changing the position of lines and columns
- Maintain performance by avoiding “search”

Rules
1. The start of a range shall be defined by a named cell
2. Columns and lines within the range shall be defined as constants

Example:
```
Dim FirstLine as integer ‘ first line of objectlist
Const ColAliasCtrl = 2          ‘Column with “Alias controller”

‘ get the first line of the objectlist
        FirstLine = Worksheets(“Spec-Device”).range(“StartList”).row + 1
‘ get the value of the first Alias Controller
        Wert = Worksheets(“Spec-Device”).range(cells(FirstLine, ColAliasCtrl)).value
```

4.5 Functional Test Specification (Req 20)

This requires splitting up the test specification into a purely functional part (workflow steps 1..4), an environment specific part (workflow steps 5..10), and the test execution (workflow steps 11..14).
Use cases:
1. Create a functional test specification (for example for QC or for IMSES)
2. Extend a functional specification with test environment data (for testing with TsNet)
3. Remove test environment data from an existing TsNet test

(Replacing a test environment is executing use case 3 and use case 2.)

4.5.1 Alias Names

Instead of BACnet object names, the test specification uses “Alias” names for BACnet objects. With using Alias names, the specification is independent of the engineering environment.

5. Description of Functions and Data

5.1 Sheet Spec-Objects

Required sheet with fix name.

Used for the definition of the BACnet objects used in the test sheets.
Goal:

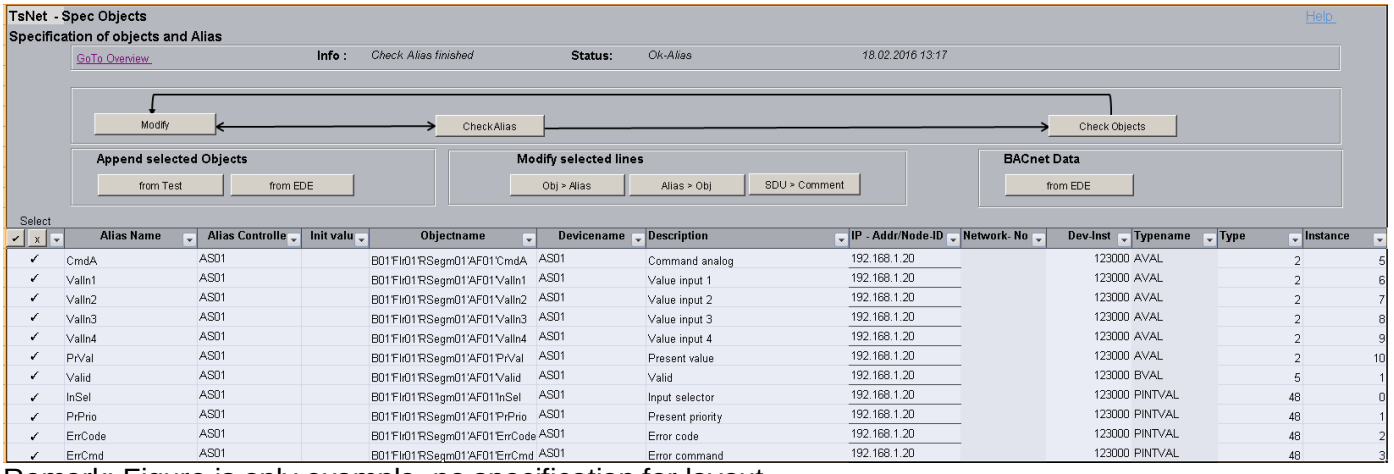
1. Define BACnet objects that will be available for test specification.
2. Provide all object data required for test execution.
3. Handling of object data from BACnet EDE list[1)].
4. Handling of workflow including data consistency checks

[1)] Object data is taken from EDE sheet. Importing EDE list (standardized *.csv file) into EDE sheet is already realized and not part of the IPA.

The upper part of the sheet allows calling the functions and shows status information (general), the lower part of the sheet allows to enter the BACnet objects required for the test (object list).

Note: The texts of labels, buttons, status and error information etc. used in the following specification should be treated as recommendation. They may be changed if it's helpful to improve usability, layout or prevents from misunderstanding.

5.1.1 Example



Alias Name	Alias Controlle	Init valu	Objectname	Devicename	Description	IP - Addr/Node-ID	Network- No	Dev-Inst	Typename	Type	Instance
CmdA	AS01		B01'Flt01'RSegm01'AF01'CmdA	AS01	Command analog	192.168.1.20		123000	AVAL	2	5
ValIn1	AS01		B01'Flt01'RSegm01'AF01'ValIn1	AS01	Value input 1	192.168.1.20		123000	AVAL	2	6
ValIn2	AS01		B01'Flt01'RSegm01'AF01'ValIn2	AS01	Value input 2	192.168.1.20		123000	AVAL	2	7
ValIn3	AS01		B01'Flt01'RSegm01'AF01'ValIn3	AS01	Value input 3	192.168.1.20		123000	AVAL	2	8
ValIn4	AS01		B01'Flt01'RSegm01'AF01'ValIn4	AS01	Value input 4	192.168.1.20		123000	AVAL	2	9
PrVal	AS01		B01'Flt01'RSegm01'AF01'PrVal	AS01	Present value	192.168.1.20		123000	AVAL	2	10
Valid	AS01		B01'Flt01'RSegm01'AF01'Valid	AS01	Valid	192.168.1.20		123000	BVAL	5	1
InSel	AS01		B01'Flt01'RSegm01'AF01'InSel	AS01	Input selector	192.168.1.20		123000	PINTVAL	48	0
PrPrio	AS01		B01'Flt01'RSegm01'AF01'PrPrio	AS01	Present priority	192.168.1.20		123000	PINTVAL	48	1
ErrCode	AS01		B01'Flt01'RSegm01'AF01'ErrCode	AS01	Error code	192.168.1.20		123000	PINTVAL	48	2
ErrCmd	AS01		B01'Flt01'RSegm01'AF01'ErrCmd	AS01	Error command	192.168.1.20		123000	PINTVAL	48	3

Remark: Figure is only example, no specification for layout.

5.1.2 Workflow

Functional specification

1. Empty sheet (no entries in object list) or modification of existing object list.
2. [Modify] → *Status*: "Working"
3. Define *Alias Name* or *Objectname* or both. One line for each object used.
4. Optional: If only *Objectname* is defined, use [Obj > Alias] to create *Alias Name*.
5. [Check Alias]
   Depending on the result:
   a. Check failed: *Info* shows reason of last failure. *Status* is "Error".
      Correct entries and re-check with [Check Alias]
   b. Check passed: *Info*: "Check Alias OK" *Status*: "OK-Alias"

i. Leave worksheet and define devices or test steps.
ii. Continue with defining more object aliases (→ Step 2)
iii. Continue with the definition of test environment (→ Step 7)

Alternative

6. Instead of steps 3..5 use [Append from Test] or [Append from EDE] to import existing object data.

Definition of the test environment

7. Define *Alias Controller* for each object.
8. Define *Objectname* for each object or use [Alias > Obj]
9. Optional: Generate *Description* with [SDU > Comment]
10. Use [from EDE] to import BACnet data from sheet EDE.

Verify data

11. [Check Objects]
    Depending on the result:
    a. Check failed: *Status Object* shows reason why check failed or "OK". *Info* shows reason of last failure. *Status* is "Error".
        i. Correct entries and re-check with [Check Objects]
        ii. Leave the worksheet and define devices or test steps.
    b. Check passed: All *Status Object* are "OK". *Status*: "OK-Ready". *Info*: "Check objects OK".
        i. Continue with adding more objects (→ Step 2)
        ii. Leave the worksheet and define test steps or start testing.

5.1.3 Workflow Support

"Modify" button is selectable, unless Status is "Working"
If "Modify" is selected, Status goes to "Working". Only "Working" allows user input. In all other states, user input is not possible
"Append selected Objects" , "Modify selected lines" and "BACnet Data" are selectable in status "working". They do not influence the status.
"Check Alias" is always selectable.
If "Check Alias" is selected, status goes to "OK-Alias" or "Error-Alias"
"Check Objects" is always selectable.
If "Check Objects" is selected, status goes to "OK-Objects", "Error-Alias" or "Error-Objects"

5.1.4 Functions

Function	Call	Affected data	Description
Modify	Command button	Complete sheet	Switches the sheet from read-only to read-write. Enables all input fields and all functions. Sets *Status* to "Working", and disables "Modify" button. Enables buttons "Append selected Objects" , "Modify selected lines" and "BACnet Data". Input fields are of type "W" in table 5.1 The background of input fields is set to white, the background of other fields are set to grey. The complete object list is set to read/write, also the grey fields.
Check Alias	Command button	Complete sheet	Checks the data and the consistency of the following input fields:

			Alias Name: Unique and not empty *Alias Controller*: Must match *Alias Controller* from Spec-Device sheet. Empty fields are not allowed. If all checks OK: Switches sheet to read-only *Status*: OK-Alias Else Switches sheet to read-only Incorrect entries are marked red. *Info* = error reason *Status*: Error-Alias. After Check alias is executed *Status date/time* shall be actualized. Enables "Modify" Disables "Append selected Objects" , "Modify selected lines" and "BACnet Data" The check itself is already implemented and not part of IPA
Append from Test	Command button	Appends new lines	Opens a dialog, asks for a test step. After confirmation, appends Alias names from selected objects in test step to the end of the list. Executes [Check Alias]. Executes [Alias > Obj] for all new entries. Not to be realized in IPA
Append from EDE	Command button	Appends new lines	Opens a dialog. After confirmation: appends object name from all selected lines of EDE list. See detailed description. Executes [Obj > Alias] for the new lines. Executes [Check Alias].
Object → Alias	Command button	Alias Name in selected lines	Reduces the *Objectname* to the short name, and copies it to *Alias Name*. If *Alias Name* is not empty, user must confirm. See detailed description. Executes [Check Alias].
Alias → Objectname	Command button	Objectname in selected lines	Adds a "*" to the *Alias Name* and copies it to the *Objectname*. If *Objectname* is not empty, then the user must confirm. See dialog boxes.
SDU to Comment	Command button	Description in selected lines	Searches for *Alias Name* in SDU and writes assigned description text to *Description.* If existing, adds *Alias Controller* in "()". If *Description* field is not empty, user must confirm. Not to be realized in IPA
BACnet Data	Command button	Columns BACnet Data	Execute [Check Alias]. Aborts BACnet Data, if error. After user confirmation, deletes all BACnet data (*Controllername, IP-Addr, Dev-Inst, Network-No,Type, Instance*) Looks for the *Objectname* in the EDE of the *Alias Controller* with help of the device list. See detailed description.
Check Objects	Command	Complete	Execute [Check Alias]. Aborts Check objects,

	button	sheet	if error. Checks the data and the consistence of all input fields, device list and EDE. See detailed description. If all checks OK: Switches sheet to read-only *Status*: OK-Ready Else Switches sheet to read-only *Status*: Error-Objects or Error-Alias Enables “Modify” Disables “Append selected Objects” , “Modify selected lines” and “BACnet Data”
Select Button	Command button	Column A	Selects line in object list and marks it with a tick (existing VBA code)
Deselect Button	Command button	Column A	Deselects line in object list and removes the tick

Table 5–1: Functions

5.1.5 Fields

The upper part of the sheet contains general data for all devices

Name	Type	Location in example	Description
Info	R	E3	Last detailed status or error information from any functions called from this sheet
*Status**	R	G3	Status of the complete sheet Working: in work, not all entries / checks done yet Error: Check Objects or BACnet Data or Check Alias failed OK-Alias: Check Alias correct OK-BACnet: BACnet Data correct OK-ready : ready for testing, all checks done
	R	H3	Date and time of the last status change
*StartList**	R	A13	One line above the start of the object list

* named cells

Table 5–2: Fields_general

The lower part of the sheet from one line below “StartList” downwards contains the object list. The object list ends with the first empty line.

Name	Type	Location in example	Description
Select	R	A13…Axx	Shows, if an object is selected for further functions (delete…). Handling via select / deselect buttons
Alias Name	W	B13…Bxx	Alias name for functional specification
Alias Controller	W	C13…Cxx	Alias controller name for functional specification
Init Value	W	D13…Dxx	Initial value of objects present value. Used when generating Init Scripts
Objectname	W	E13…Exx	Object name, reference to EDE list
Device	R	F13…Fxx	BACnet controller name for definition of the actual test

name			environment
Description	W	G13…Gxx	Description of the Object (for documentation only)
IP-Addr	R	H13…Hxx	IP-Address of the controller or Node-ID for MSTP / LON
Network-No	R	I13…Ixx	Network-Number for communication via IP/MSTP or IP/LON-Router
Device Instance	R	J13…Jxx	BACnet device instance
TypeNum	R	K13…Kxx	BACnet object type, numeric value for communication
TypeAsc	R	L13…Lxx	BACnet object type, short name for documentation
Instance	R	M13…Mxx	BACnet object instance for communication
Status Object	R	N13…Nxx	Status of the object, last error / status message Obj not unique : from function Check Alias / Check objects / BACnet data Obj not found : Object not found in EDE of the device Dev not found: Device not found in EDE or in Spec-Devices No entry: OK

Table 5–3: Fields_Objectlist

5.1.6 Dialog boxes

Function	Description
Append from EDE	Shows list of selected object-name from EDE Allows the user to continue appending objects or to abort the function
Object → Alias	Prompts, that the *Alias name* is not empty. Allows the user to overwrite the *Alias name* or not to do so. A tick box can be selected to do the same with all other *Alias* name.
Alias → Objectname	Prompts, that the *Object name* is not empty. Allows the user to overwrite the *Object name* or not to do so. A tick box can be selected to do the same with all other *Object* name.

Table 5–4: Dialog_Boxes

5.1.7 Detailled description

5.1.7.1 Append from EDE

1. De-select all lines in objectlist.
2. With all lines in EDE, which are selected.
3. Append a line to the object list in Spec-Objects
4. Copy column *object-name* from sheet EDE to column *Objectname* in Spec-Objects
5. Copy column *description* from sheet EDE to column *Description* in Spec-Objects
6. See, if *dev obj.id* from sheet EDE can be found in column *Dev-Inst* of sheet Spec-Devices. If found, then
    a. copy *Alias controller* from this line in sheet Spec-Devices to column *Alias controller* in Spec-Objects
    b. copy *Device-Name* from this line in sheet Spec-Devices to column *Device name* in Spec-Objects

    else leave the fields empty.
7. Select the newly appended line

5.1.7.2 Object → Alias

1. With all lines in *objectlist* which are selected
2. Check, if Alias Name is empty. If not, ask the user for overwriting *Alias Name* (see dialog boxes).
3. Use the *Object name*. Look for the last ' in the objectname, and take the part of the string following this position. Example: Object name = B01'Flr01'RSegm01'AF01'CmdA. Result = CmdA. Take the result and write it to *Alias Name*.

5.1.7.3 BACnet data

1. With all lines of the objectlist
2. See, if *Alias Controller* from sheet Spec-Objects can be found in column *alias Controller* of sheet Spec-Devices. If found, then
    a. copy *Device-name* from this line in sheet Spec-Devices to column *Device name* in Spec-Objects
    b. copy *IP-Addr Node-ID* from this line in sheet Spec-Devices to column *IP-Addr* in Spec-Objects
    c. copy *Network-No* from this line in sheet Spec-Devices to column *Network-No* in Spec-Objects

   d. copy *Dev-Inst* from this line in sheet Spec-Devices to column *Dev-Inst* in Spec-Objects

else leave the fields empty.

3. See, if *Objectname* from sheet Spec-Objects can be found in column object-name of sheet EDE and if *Dev-Inst* from sheet Spec-Objects matches to *Dev-Inst* of sheet EDE. If found, then
   a. Copy *object-type* from sheet EDE to *Type* from sheet Spec-Objects.
   b. Copy *object-instance* from sheet EDE to *Instance* from sheet Spec-Objects
   c. Look, if *Object-type* from sheet EDE can be found sheet Enum, where Column a = "BACnet_Objecttype" and column D = *Object-type* from sheet EDE.
      If found, look for column D, and copy this value to *Typename* of sheet Spec-Objects.

   Else leave the fields empty.

5.1.7.4 Check objects

1. With all lines of objectlist
2. ~~See, if *Alias Controller* from sheet Spec-Objects can be found in column *alias Controller* of sheet Spec-Devices~~. If found, then
   a. ~~If *Device-name* from this line in sheet Spec-Devices is not equal to column *Device-name* in Spec-Objects → Check objects failed~~
   b. ~~If *IP-Addr Node-ID* from this line in sheet Spec-Devices is not equal to column *IP-Addr* in Spec-Objects → Check objects failed~~
   c. ~~if *Network-No* from this line in sheet Spec-Devices is not equal to column *Network-No* in Spec-Objects → Check objects failed~~
   d. ~~if *Dev-Inst* from this line in sheet Spec-Devices is not equal to column *Dev-Inst* in Spec-Objects → Check objects failed~~
3. ~~If Check objects failed, then~~
   a. ~~Mark the failed entry (*Device-name* or *IP-Addr* or *Network-No* or *Dev-Inst)* in red.~~
   b. ~~Set *Status Object* to "Error device"~~
   c. ~~Depending on the reason, Set *Info* to "Inconsistent <Reason> here and in Spec-Devices" (<Reason> see a.)~~
4. ~~See, if *Objectname* from sheet Spec-Objects can be found in column object-name of sheet EDE and if *Dev-Inst* from sheet Spec-Objects matches to *Dev-Inst* of sheet EDE. If found, then~~
   a. ~~If *object type* from sheet EDE is not equal to *Type* from sheet Spec-Objects → Check objects failed~~
   b. ~~If *object instance* from sheet EDE is not equal to *Instance* from sheet Spec-Objects → Check objects failed~~

   Else Check objects failed.
5. ~~If Check objects failed, then~~
   a. ~~Mark the failed entry (*Object name* or *Type* or *Instance)* in red.~~
   b. ~~Set *Status Object* to "Error Objects"~~
   c. ~~Depending on the reason, Set *Info* to "Inconsistent <Reason> here and in EDE" (<Reason> see a.)~~
6. Look, if *Object-type* from sheet EDE can be found sheet Enum, where Column a = "BACnet_Objecttype" and column D = *Object-type* from sheet EDE.
   If found, look for column D, and copy this value to *Typename* of sheet Spec-Objects.
   else Check objects failed
   a. Mark the failed *type* in red.
   b. Set *Status Object* to "Error Objects"
   c. Set *info* to "Object type not supported"