The Source of Your Success...

QIP4-Testing

User's Manual

V2.2

Jan Boes

23-8-2012

QIPtree B.V. • Van Voordenpark 1A • 5301 KP Zaltbommel • The Netherlands • +31 (0)418 68 35 00 • www.qiptree.com
Bank: 067658716 • KvK: 17111736

Index

1 Introduction

This document describes the QIP4-Testing product version 2.2.

During the programming and testing phase of a functional design, the NAV functionality is tested extensively, first during the programming phase by the developer and later during the test phase by the consultant. Often the same functionality is tested and / or exceptions are overseen.
To speed up the process the QIP4-Testing has been developed.
This tool is able to test the NAV processes by testing the expected outcome of these processes.
The outcome of the processes can be tested via:

- Determine if error message(s) appear in the process
- Are expected values met after the process has been executed

1.1 A schematic overview of the whole process

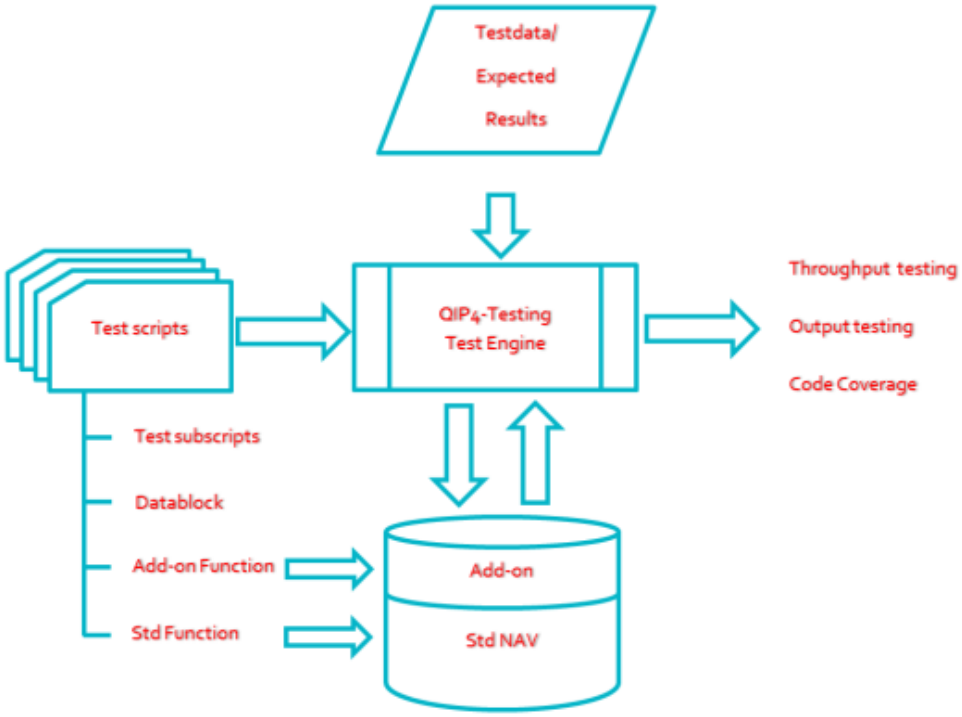


Figure 1: Schematic overview of QIP4-Testing

The test sets and the expected values are created by the user. Then the test is executed by running the standard or customized Microsoft Dynamics NAV processes and in the end the user can check the test results to see if the NAV process has run correctly

In the following chapters the standard Microsoft Dynamics NAV testability features and the QIP4-Testing features are described.

2 Microsoft Dynamics NAV testability features

In Microsoft Dynamics NAV 2009 SP1 the NAV testability features where introduced. The features are built in the NAV executables and NAV objects were created to test the standard NAV process.

2.1 Test suite

With a test suite it became possible to execute test without any user interference. This means that specific functions where created to mimic the behavior of the user action and in case errors occurs the information will be written to a test log

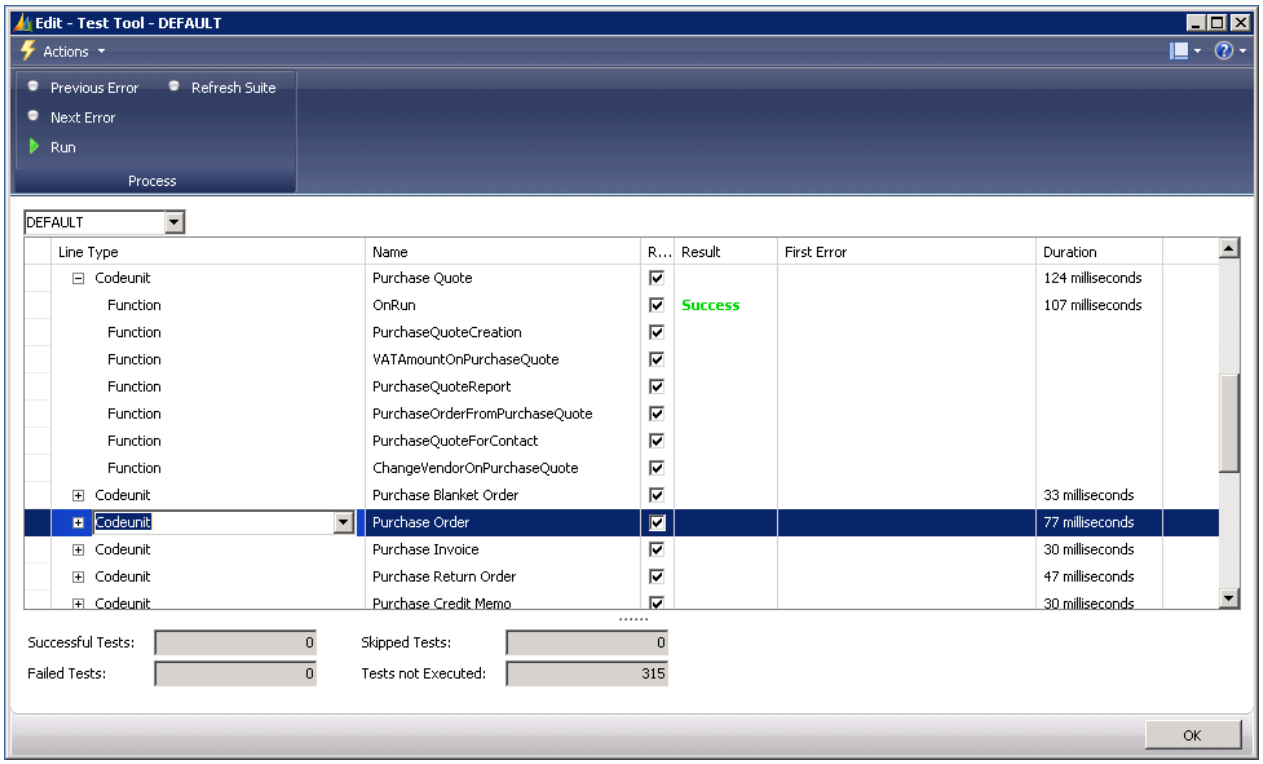


Figure 2: Test tool overview

More information can be found on the MSDN website at http://msdn.microsoft.com/en-us/library/ee414224.aspx.

2.2 Integration between the NAV testability features and the QIP4-Testing

A new codeunit with the NAV Testability Features option "Test" is added. This codeunit is the starting point for the QIP4-Testing add-on.

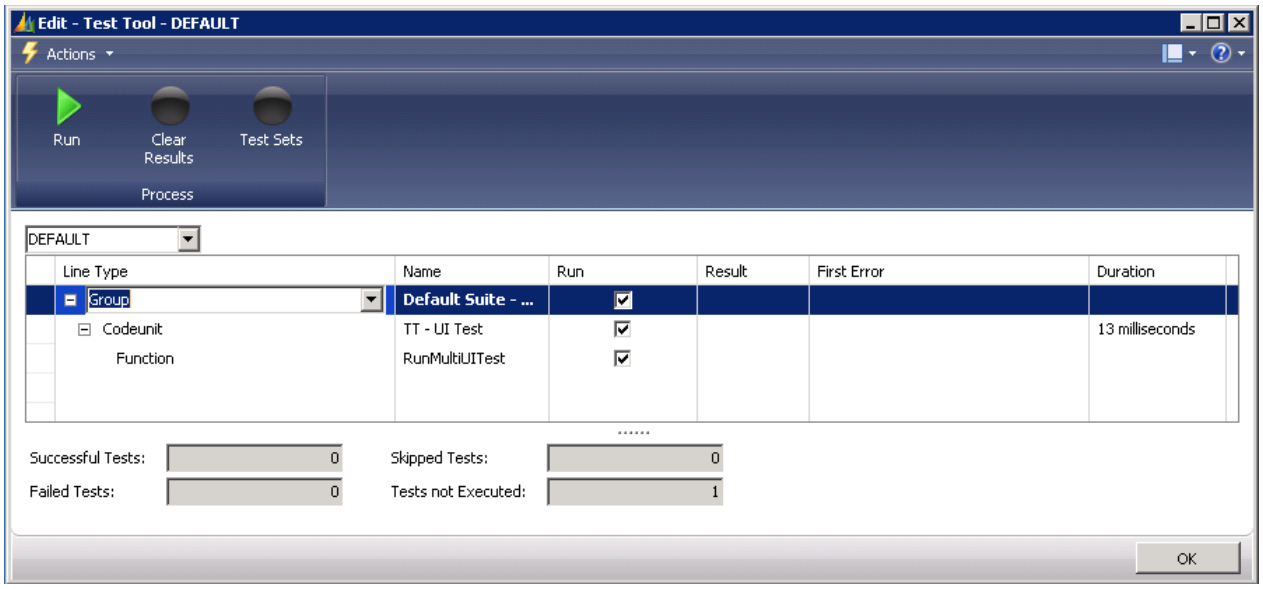


Figure 3: Test tool overview with the newly added codeunit "TT – UI Test".

Field	Description	
Line Type	Group	A group of test codeunits
	Codeunit	A specific test codeunit
	Functions	A test function in a test codeunit
Name	Name of the specific group, codeunit or function	
Run	In case a check mark is placed, the test will be executed	
Result	The result of the test will be displayed	
First error	In case the test failed, the first error will be displayed	
Duration	The time it took to execute the test	

A new function "UI Test Sets" is added to the Test Suite page and is displayed page is used to determine which tests have to be executed.

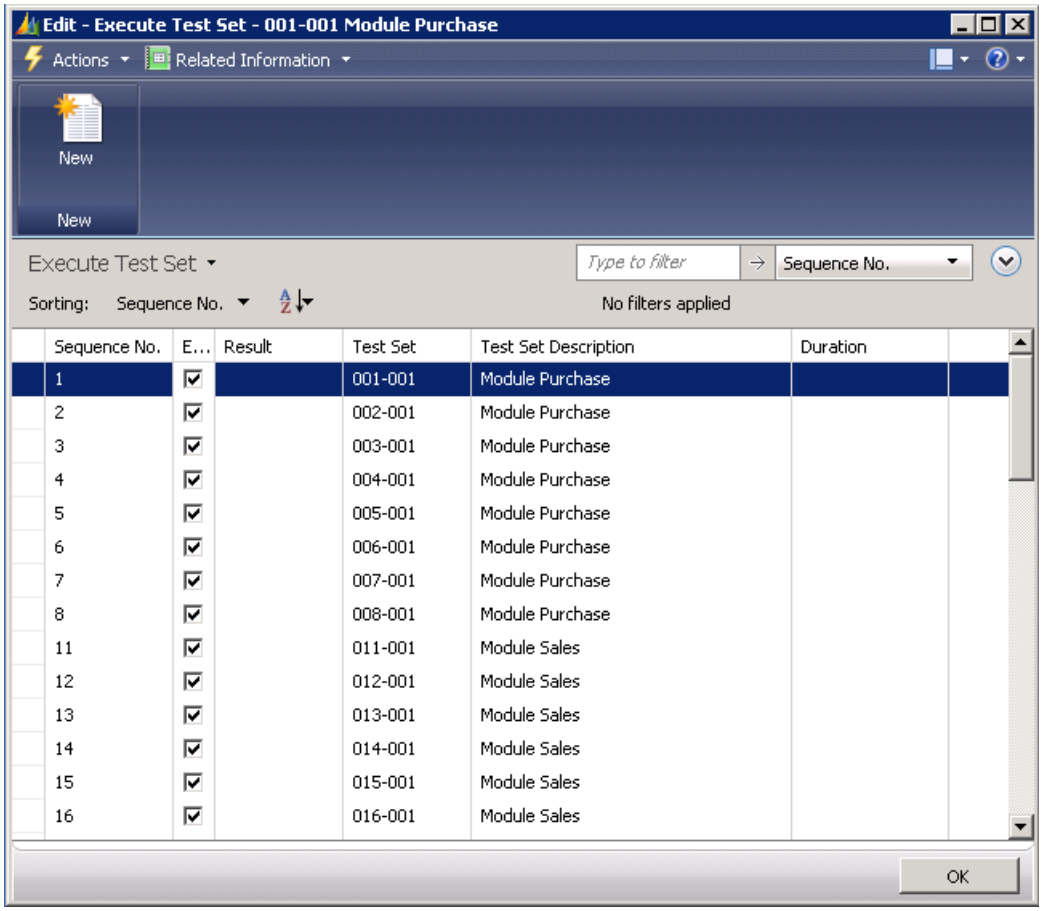


Figure 4: Examples of different test sets

Field	Description
Sequence No.	The tests are executed in the sequential order
Execute	In case a check mark is placed, the test will be executed
Test Set	The unique code of the test set
Test Set Description	Description of the test set
Duration	The time it took to execute the test set

3 QIP4-Testing functionality

There are three main tasks that can be executed by using the QIP4-Testing.

- Check execution of all code lines
- Happy flow testing
- Expected values testing

Check execution of all code lines means that the standard NAV Code Coverage tool is started before the test sets are executed. After the execution the Code Coverage tool is stopped. In the Code Coverage tool an overview is visible if all code lines are executed.

Happy flow testing means that you test one or more NAV processes to make sure these are (still) working after modifications have been made.

Expected value testing means that you test if one or more fields have the expected value(s) after one or more processes are executed.

In all cases a test set is needed for these tests. This test set is the main object in the QIP4-Testing. A test set can consists of data, functions or another test set.

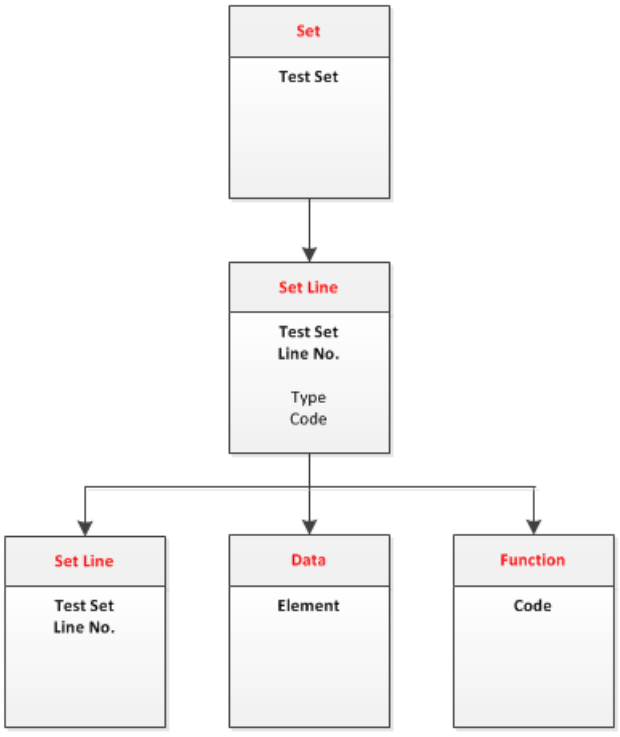


Figure 5: Test set functional overview

Data:
Data is used to manipulate NAV data. For instance inserting a purchase order header or finding the last reservation entry.

Function:
A specific NAV function that should be executed. For example posting a purchase order or archiving a sales order.

Set Line:
A test set can be added as a subset to another test set. For instance creating inventory can be a sub test set when the sales functionality is tested.

3.1 Data blocks

As already mentioned data can be used to manipulate NAV data. The page window consists of two parts. In the header the table, key and database actions are shown and in the lines the values of the fields can be inserted.

In the figure below a purchase order header is inserted. Because a record is inserted in table 38 (purchase header) and the document type is set to order.
The order will be bought by vendor 10000 and will be received in warehouse BLUE.

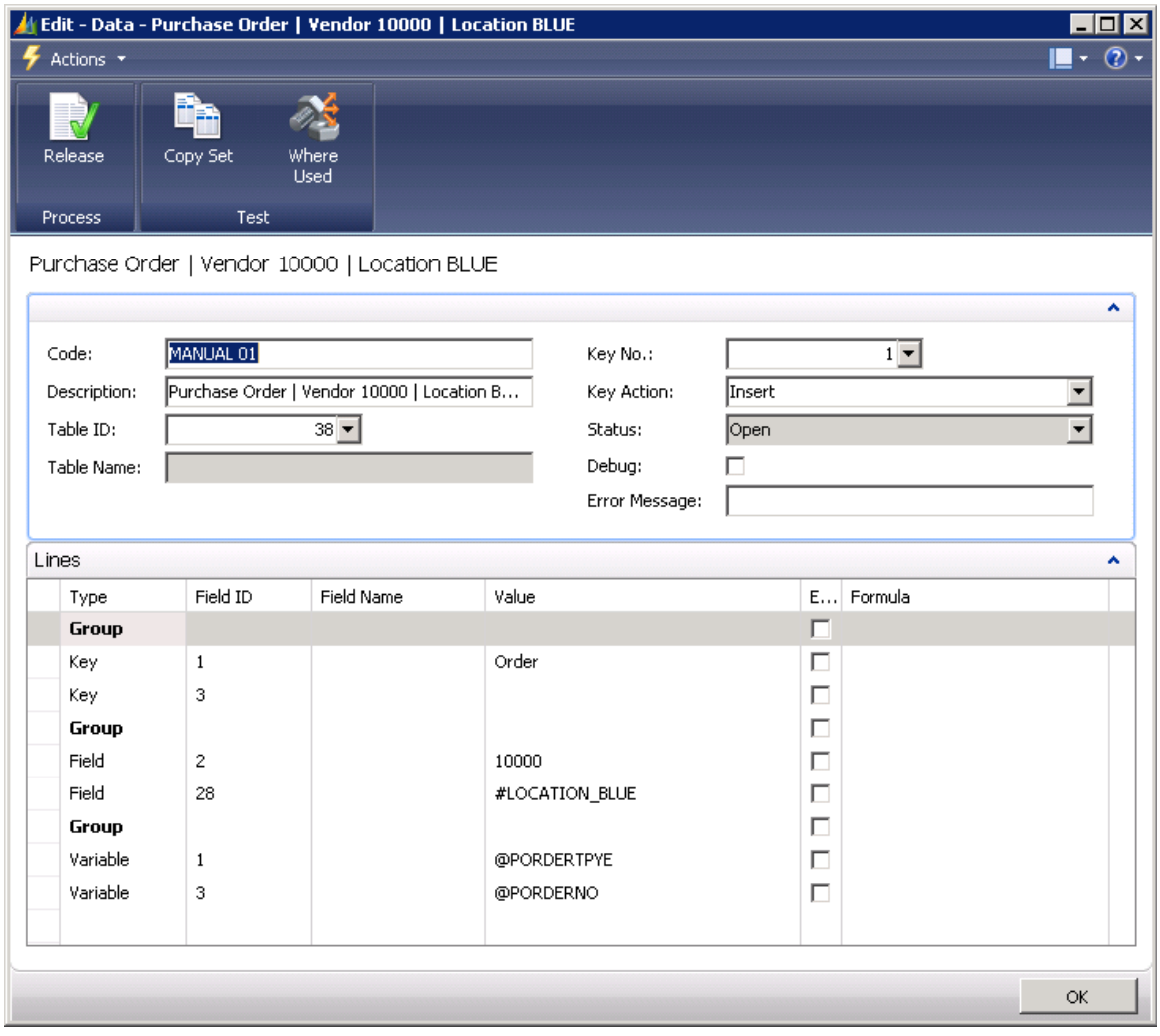


Figure 6: Creating a purchase order header

Functions

Actions		
Part	Function	Description
Process	Reopen	Reopen the data block
Test	Copy Set	Copy data block to another data block
	Where Used	Gives an overview in which test sets the data is used.

In the header the following fields are available

Field	Description	
Code	The unique code of the data	
Description	Description of the data	
Table ID	Table ID on which the manipulation should take place	
Table Name	Table name of the table	
Key No	The key which should be used by the data manipulation	
Key Action	Insert:	Insert a record in the table
	FindFirst:	Find the first record that match the filter criteria
	FindSet:	Find a set of records that match the filter criteria
	FindLast:	Find the last record that match the filter criteria
	Delete:	Delete the record(s) that match the filter criteria
	Delayed insert:	Insert the record when all fields have been filled
Status	Open	Modifications can be made in the page
	Released	No modifications can be made in the page
Debug	Sometimes the code is not working as expected. By placing a check mark in the debug field and placing a debug stop in the NAV A/L code, problems can be found faster.	
Error message	When an error is expected to happen in the data block, the expected error message can be filled here (and thus tested as expected output)	

The following fields are available in the line

Field	Description	
Type	Group	The group type is used as a header between the different types
	Key	Based on the selected key in the header, the key fields are displayed in the lines. When the key action "FindFirst, FindSet or FindLast" have been defined, the key fields act as data filters.
	Field	In the field part, the different fields are shown. The corresponding values will be inserted in the table.
	Variable	The functionality of variables will be explained below in this paragraph

Field ID	Field ID of the value that should be inserted in the table.
Field Name	Field Name of the field
Value	Value that should be inserted in the table
Error cause	When this field is activated, the system expects an error message when this field is filled via a test set.
Formula	A simple formula like +1 or +2W can be used.

3.1.1 Variables

In case the above displayed data is executed, a new purchase order is created. Because no value is filled by the field No., the standard NAV business logic is executed to determine the next purchase order number. By using variables the newly created purchase order number can be stored and used in another data block or as function parameter or by the determination of expected values.

Below an example is shown.
In the purchase order header two new variables are created to store the purchase type and purchase number. These variables are used in the purchase order line .

Purchase Order Header

Data block lines

Type	Field ID	Field Name	Value
Group		**Key**	
Group		**Field**	
Group		**Variable**	
Variable	1	Document Type	@PORDER_TYPE
Variable	3	No.	@PORDER_NO

Purchase Order Line

Data block lines

Type	Field ID	Field Name	Value
Group		**Key**	
Key	1	Document Type	@PORDER_TYPE
Key	3	Document No.	@PORDER_NO
Key	4	Line No.	
Group		**Field**	
Group		**Variable**	

To make a distinction between normal - and variable values, variable values should always start with a @.
A correct variable value is thus @PORDER_TYPE.

Special variables
There are some special variables which can be used

Vale	Description
WORKDATE	Place the current date in the field
TODAY	Place the current date in the field
?	Place a random value in the field (to be used for vendor invoice no.)
USERID	Place the USER ID in the field

3.1.2 Formula

With the formula field, you can perform simple calculations. There are two calculation types possible.

Integer
When the next reservation entry number should be used in a data block, it is hard to determine it. Based on the existing data only the last known reservation entry can be found (use a data block and find the last record in the reservation entry table).
By using the formula +1, based on the last record the next reservation entry number can be calculated.

Date
To determine the date over two weeks the formula +2W case be used.

3.1.3 Error testing

To test if an error situation will occur (like the error message "The minimum permitted value is 0 " when an negative prepayment percentage is inserted in a purchase order) two steps have to be done.

1. Activate the "Error cause" field in the data block line
2. Insert the correct error message in the data block header

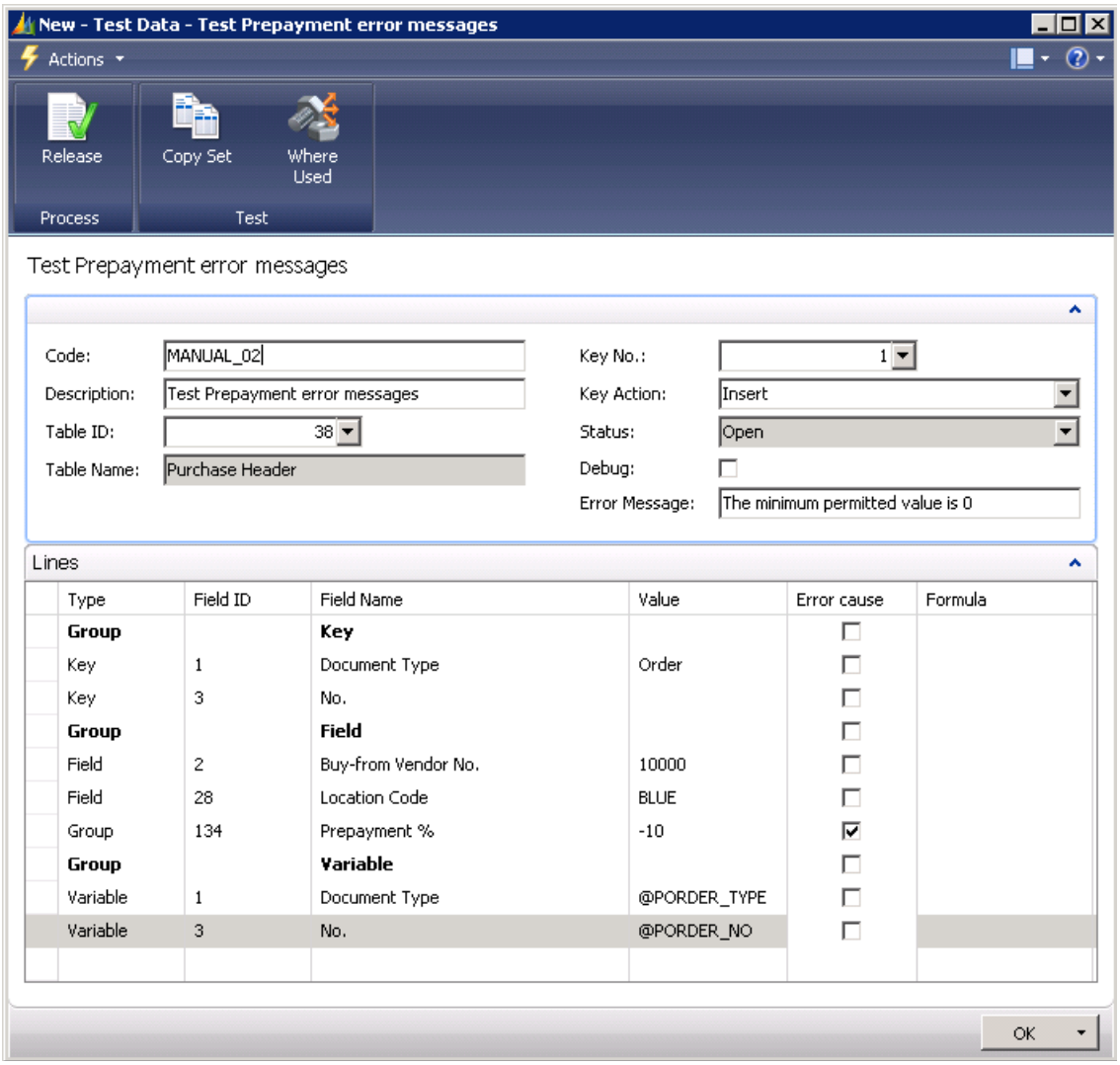


Figure 7: Testing an error situation when data is inserted

3.2 Test Functions

Of course a test set does not contain only data, but it should also execute some functionality. For a number of standard NAV functions test functions have been created. These are listed below.

The following system functions are available

Function	Description
COPY FILES	Copy files to another directory
DELETE FILES	Delete files from a directory
DELETE PROCES DATA	Delete all NAV process data (setup information is not deleted)
HYPERLINK	Execute an external program
SET WORK DATE	Set a specific work date
SLEEP	NAV will stop processing and will start after the sleep time has ended

The following NAV functions are available:

Function	Description
ARCHIVE PO DOC	Archive purchase document
ARCHIVE SO DOC	Archive sales document
CALC PURCH DISC	Calculate purchase discount
CALC SALES DISC	Calculate sales discount
CALC. WHSE. INVENT.	Calculate Warehouse Physical Inventory
CONV PQ TO PO	Convert Purchase Quote to Purchase Order
COPY PO DOC	Copy purchase document
COPY SALES DOC	Copy sales document
CREATE PICK	Create Pick
CREATE PUT-AWAY	Create Put-Away
CREATE WHSE. RCPT PO	Create Whse. Rcpt. from Purchase Order
CREATE WHSE. RCPT SO	Create Whse. Rcpt. from Sales Order
CREATE WHSE. RCPT TO	Create Whse. Rcpt. from Transfer Order
CREATE WHSE. SHPT PO	Create Whse. Shpt. from Purchase Order
CREATE WHSE. SHPT SO	Create Whse. Shpt. from Sales Order
CREATE WHSE. SHPT TO	Create Whse. Shpt. from Transfer Order
GET RCPT LINES	Get receipt lines
GET STD. PURCH CODES	Get Standard Purchase codes
GET STD. SALES CODES	Get Standard Sales codes
INVOICE PO	Invoice Purchase Order
INVOICE SO	Invoice Sales Order
PO CANCEL APPR. REQ.	Cancel Purchase Order Approval Request
PO EXPLODE BOM D.BOM	Explode Purchase Order Bill of Material – Dimensions from BOM
PO EXPLODE BOM D.COM	Explode Purchase Order Bill of Material – Dimensions from Components
PO INSERT EXT. TEXT	Insert Extended Texts for Purchase Order
PO SEND APPR. REQ.	Send Purchase Order Approval Request
POST NOTHING FOR PO	Post nothing for Purchase Order
POST NOTHING FOR SO	Post nothing for Sales Order
POST WHSE. RCPT	Post Warehouse Receipt
POST WHSE. SHPT	Post Warehouse Shipment
POST WHSE. SHPT + INV	Post and Invoice Warehouse Shipment
REC. AND INV. PO	Receive and Invoice Purchase Order
RECEIVE PO	Receive Purchase Order
REFRESH PROD. ORDER	Refresh Proudction Order
REGISTER WHSE ACTIV.	Register Whse. Activity Doc.
RELEASE PO	Release Purchase Order

RELEASE SO	Release Sales Order
RELEASE TO	Release Transfer Order
RELEASE WHSE. SHPT	Release Warehouse Shipment
REOPEN PO	Reopen Purchase Order
REOPEN SO	Reopen Sales Order
REOPEN TO	Reopen Transfer Order
REOPEN WHSE. SHPT	Reopen Warehouse Shipment
SHIP AND INV. SO	Ship and Invoice Sales Order
SHIP SO	Ship Sales Order
SO CANCEL APPR. REQ.	Cancel Sales Order Approval Request
SO EXPLODE BOM	Explode Sales Order Bill of Material
SO INSERT EXT. TEXT	Insert Extended Texts for Sales Order
SO SEND APPR. REQUEST	Send Sales Order Approval Request

Note: Most of these functions have parameters so a specific function can be connected to a specific order.

These functions are available in the QIP4-Testing function library. The page can be found via Departments – Testing – Test Manager – Test Functions.

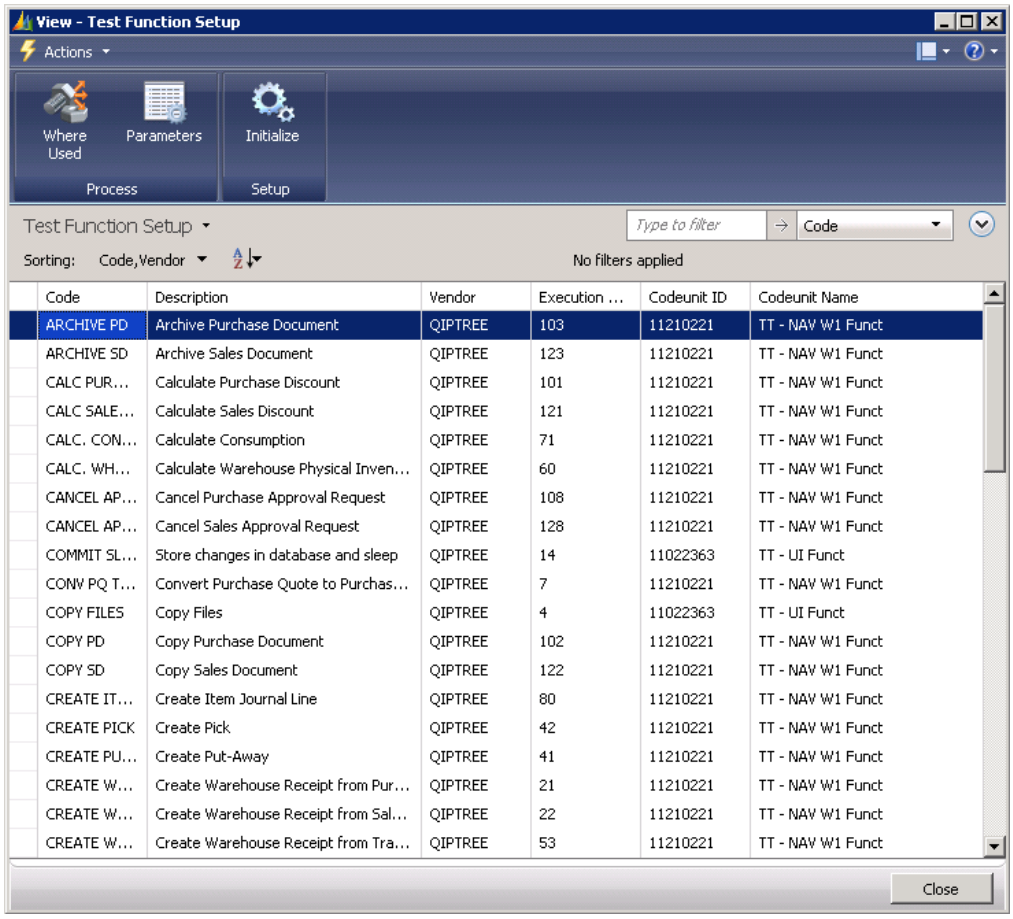


Figure 8: Test functions overview.

The following fields are available

Field	Description
Code	The unique code of the function
Description	Description of the function
Vendor	Vendor who created the function
Execution ID	Internal ID number to identify the function
Codeunit ID	NAV Codeunit where the function C/AL code is located
Codeunit Name	Name of the NAV Codeunit

When the Microsoft Dynamics NAV fob file is imported in the database these functions are not yet available. By pressing the Initialize button the functions will be generated and also the vendors of these functions will be listed in the vendor overview.

This page can be found via Departments – Testing – Setup – Vendor

Figure 9 : Vendor overview

To see which specific Test codeunits have been created by the vendor the Codeunits button can be selected.

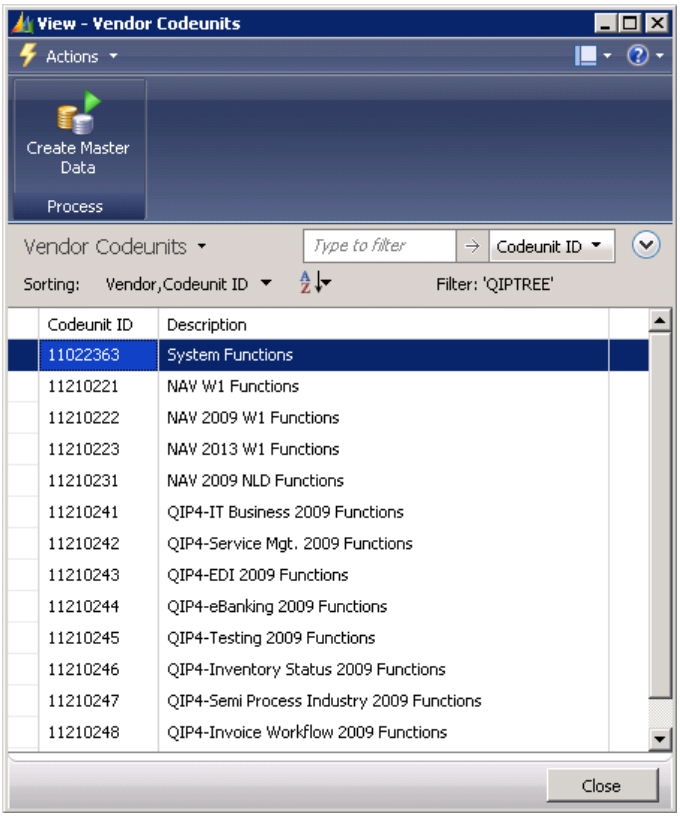


Figure 10: Overview of the vendor codeunits.

The button "Create Master Data" can be used to add setup data to the NAV database. How this C/AL code must be written is explained in chapter 4.

3.3 Test set

This paragraph describes the creation and execution of test sets. In the setup is determined in which NAV company these test sets can be executed.

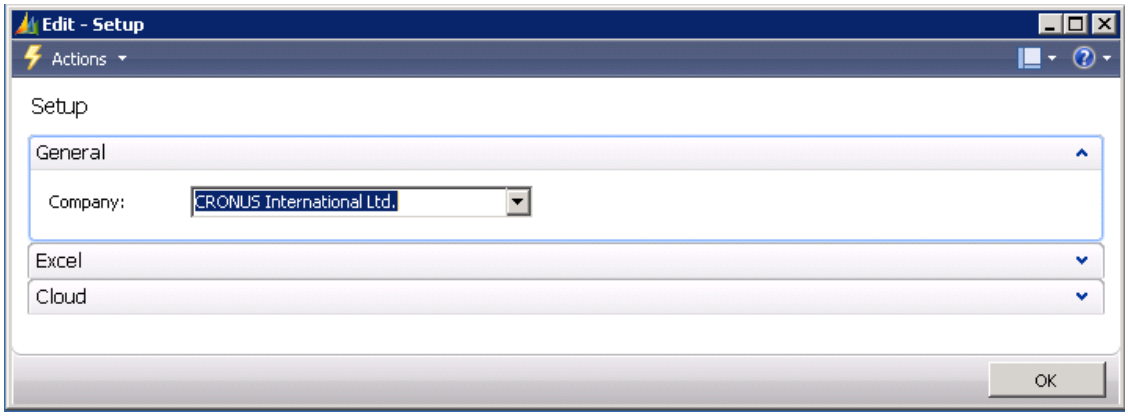


Figure 11: QIP4-Testing setup information

As already mentioned data and functions can be combined into a test set. By doing so a standard and / or customized NAV process (like receiving a purchase order) can be tested.

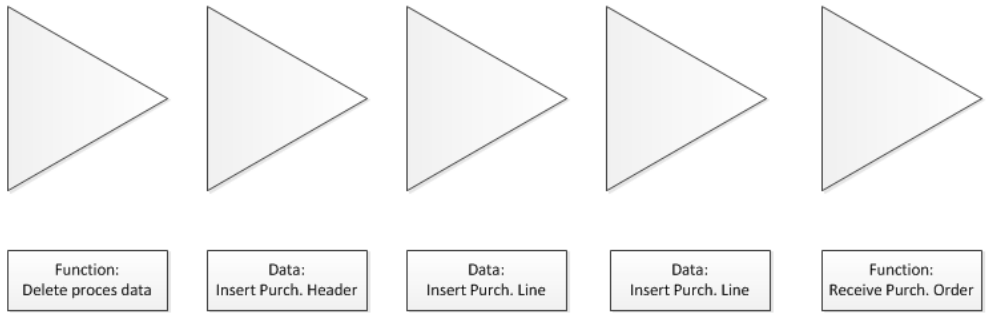


Figure 12: Test steps to create a complete test set

A set has to be created in four steps

1. Create data blocks
2. Create the test set
3. Connect data blocks to the set
4. Connect functions to the set and fill function parameters

To understand the process of creating a set in detail a walk through chapter is added to this manual.

3.3.1 Create data blocks

The creation of data blocks is already explained in paragraph 3.1.

3.3.2 Create test set

The page consists of two parts. In the header the test set identification is shown and in the lines the different test steps.

In the figure below a test set header is inserted.

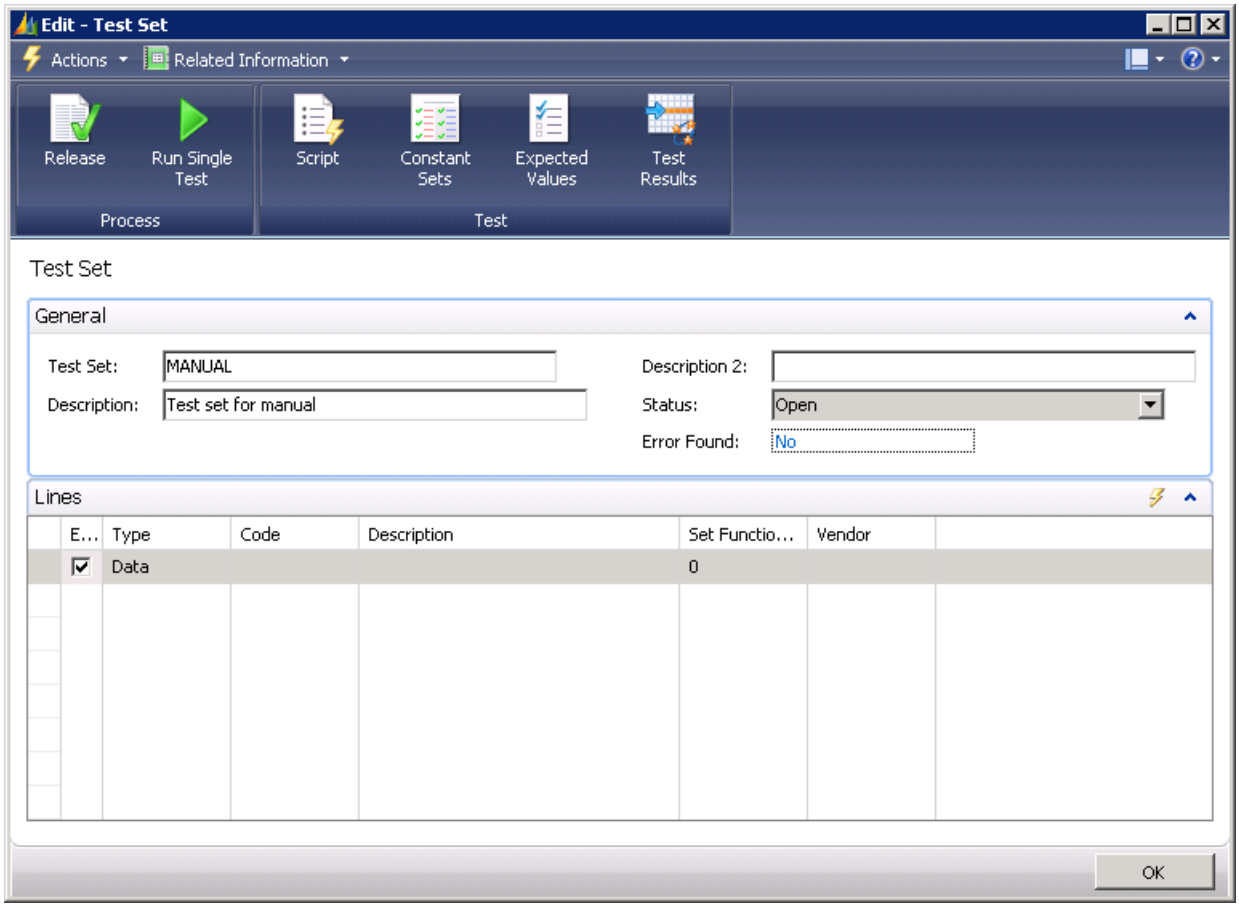


Figure 13: Test set card page

Functions

Related Information		
Part	Function	Description
Process	Release	Release the test set
	Run Single Test	Run only this test (User Interface messages will not be suppressed)
Test	Scripts	All test steps in the test set
	Constant Sets	Test constant sets
	Expected Values	Expected test values
	Test Results	Results of the expected values testing

In the header the following fields are available

Field	Description
Test Set	The unique code of the test set
Description	Description of the test set
Description 2	Description of the test set

Status	Open	Modifications can be made in the page
	Released	No modifications can be made in the page
Error Found	In case an error is found in the test set in previous test runs, the field will show the value yes	

The following fields are available in the line

Field	Description	
Execute	During development it can be really handy to execute the first lines automatically via the test set, but not the execute the function in which the error occurs. After execution of the test set the developer can start with the debug to find the error	
Type	Text	In the description field a descriptive message can be written,
	Data	A data block will be added to the test set
	Function	A function will be added to the test set
	Set	A test set will be added to the test set
Code	The identification of the data block / function / test set	
Set Function line	A function code can be added by multiple vendors. To make a correct distinction between them the function line is added	
Vendor	Vendor of the used function	

3.3.3 Set Function parameters

By using the lookup functionality of the code field the appropriate data block, function or test set can be selected. For the data block and test set no additional actions are needed, but for functions it can be needed to give the correct function parameters values.

When the lookup of the code field is selected a new page opens and in it the set functions are displayed. For a new test set no functions have been defined therefor an empty page will be displayed.

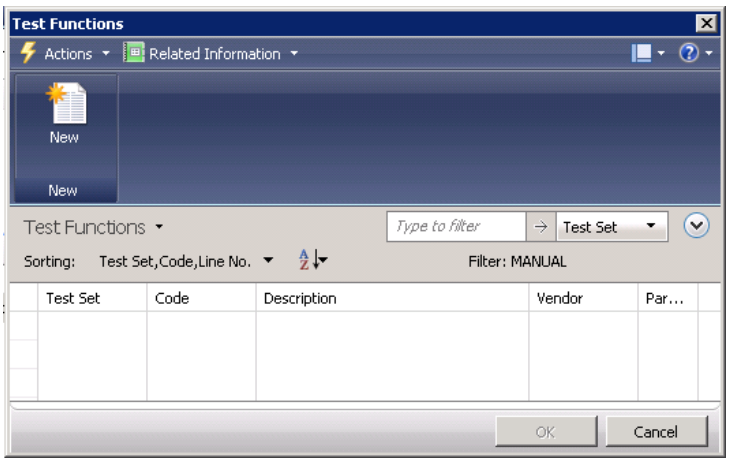


Figure 14: Test set function list page

Via the button "New" a the test set function page is opened and the appropriate function can be selected. In this manual the receiving of a purchase order is described so the function "Receive PO" should be selected. Because this function has parameters (NAV needs to know which purchase order is received) the appropriate function values should be inserted.

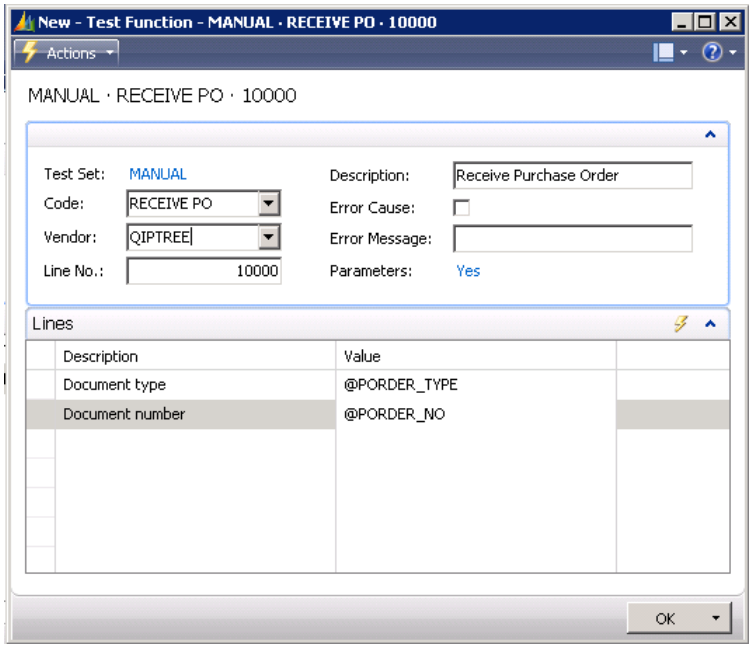


Figure 15: Test set function list page

In the header the following fields are available

Field	Description
Test Set	The unique code of the test set
Code	The unique code of the selected function
Line No.	The function line number to make a distinction between two vendor functions with the same code
Description	Description of the selected function
Error cause	When this field is activated, the system expects an error message when this function is executed
Error message	When an error is expected to happen in the function, the expected error message can be filled here (and thus tested as expected output)
Parameters	In case the function has parameters, the value will be Yes

The following fields are available in the line

Field	Description
Description	Description of the function parameter
Value	Value of the function parameters

3.3.4 Set constants

In many cases it is handy to test different input values. This can be a different NAV setup (switch options on or off, or to test different values so dimension setup or discount calculations can be checked.

In this manual the receiving of a purchase order is described. In case the receiving of a purchase quote should also be tested, two additional steps should be executed to create these tests.

1. Change the value of the document type into a constant
2. Create the constant sets.

Purchase Order Header

Data block header

Type	Field ID	Field Name	Value
Group		**Key**	
Key	1	Document Type	#ORDER
Key	3	No.	
Group		**Field**	
Group		**Variable**	
Variable	1	Document Type	@PORDER_TYPE
Variable	3	No.	@PORDER_NO

To make a distinction between normal values, variables and constants, constants have to be started with an #. A correct constant is thus #ORDER.

From the test set page, the button constant sets will open the test constant list page.

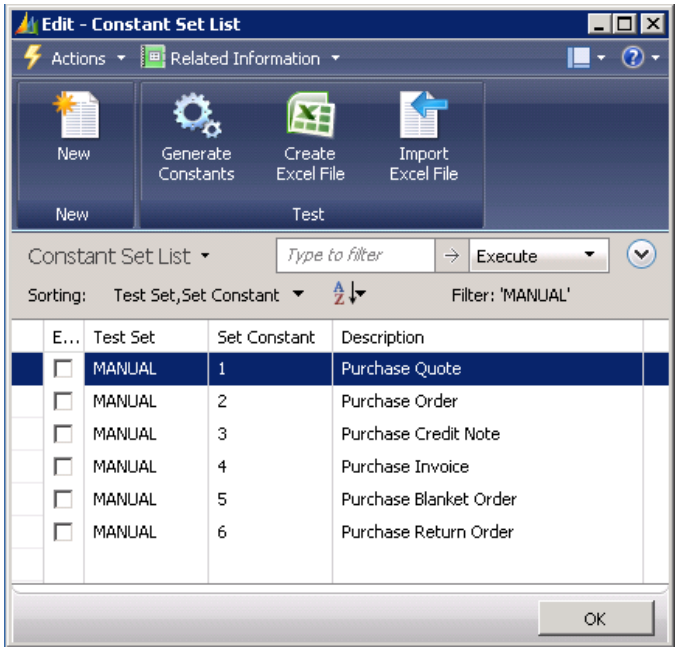


Figure 16: Set constant list page

Functions

Related Information		
Part	Function	Description
Function	Generate Test Constant	Generate the test constants
	Create Excel File	Create an Excel File Template
	Import Excel File	Import the Excel File Template

Six constant sets have been created for the different purchase document types.
Via the button "Generate Constants" all constants from the test set will be connected to the specific constant set. Via the function "Related Information" – Card the constant set card can be opened and the constant value can be added.

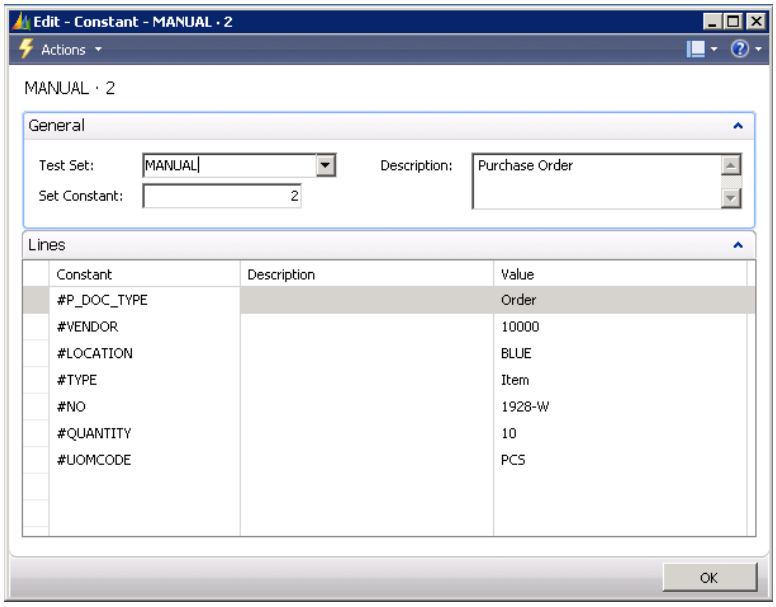


Figure 17: Set Constant Card page

When there are a lot of constants that should be filled or a huge number of constant sets are created, it is easier to use Microsoft Excel to fill in the different constant values.

When the action button "Create Excel File" is pressed an excel file is created in the specified directory in the setup page.

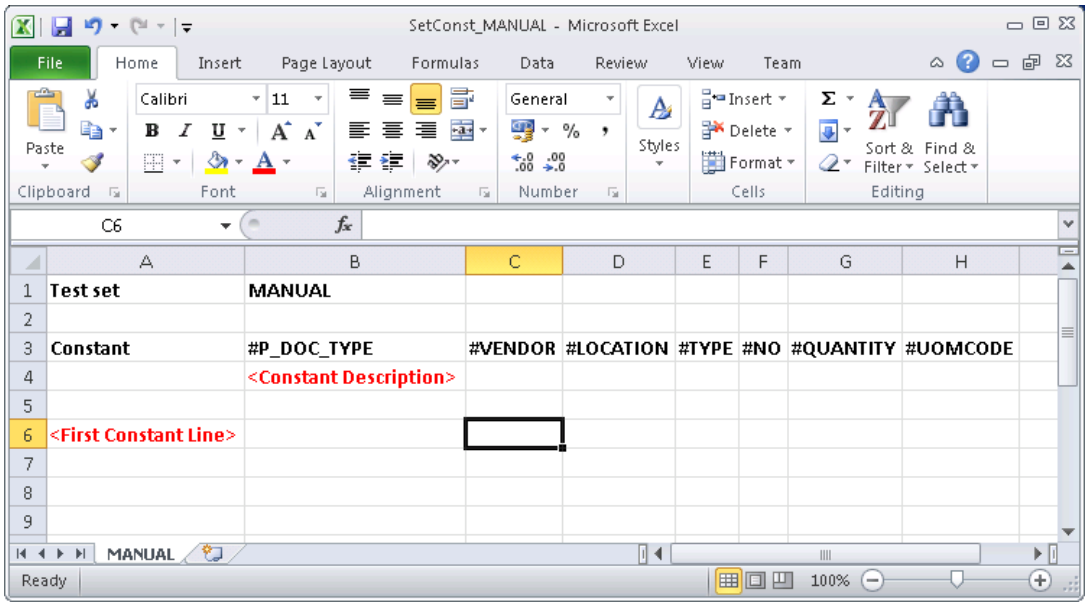


Figure 18: Excel file template to be used to add Set Constant information

On line 4 an additional description of the constant can be filled and from line 6 onward the different constant sets and values can be filled.

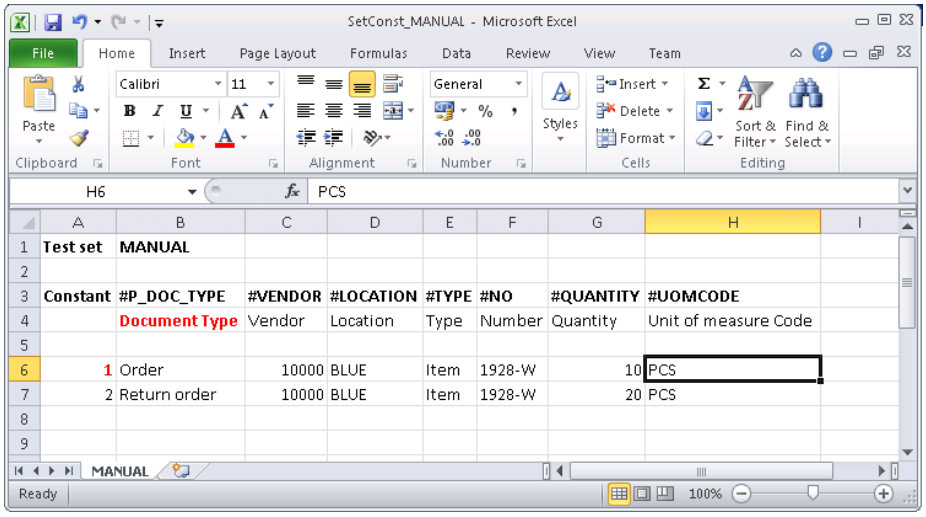


Figure 19: Excel file template filled with Set Constant information

Now the information can be imported and the constant sets and values are created.

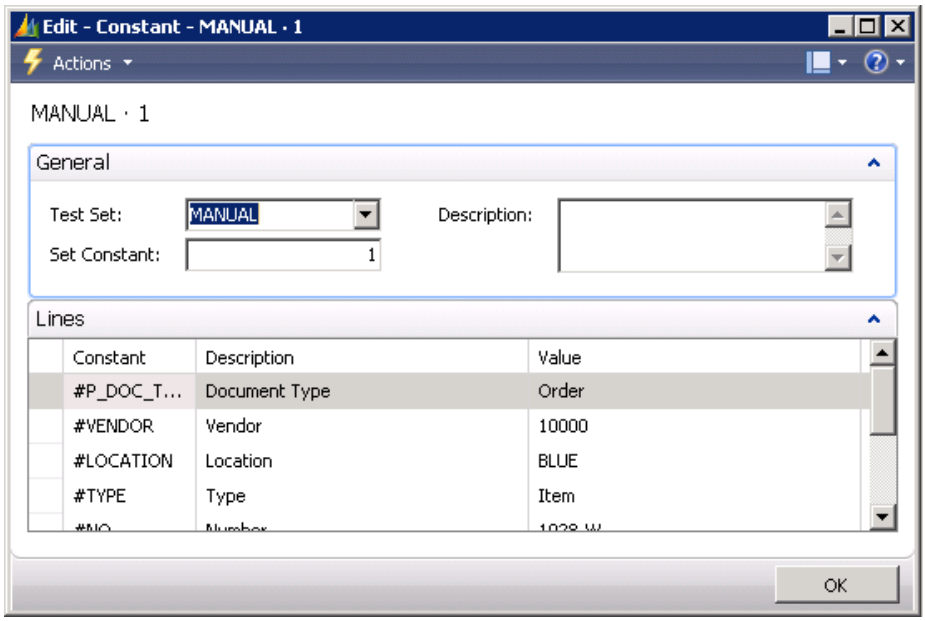
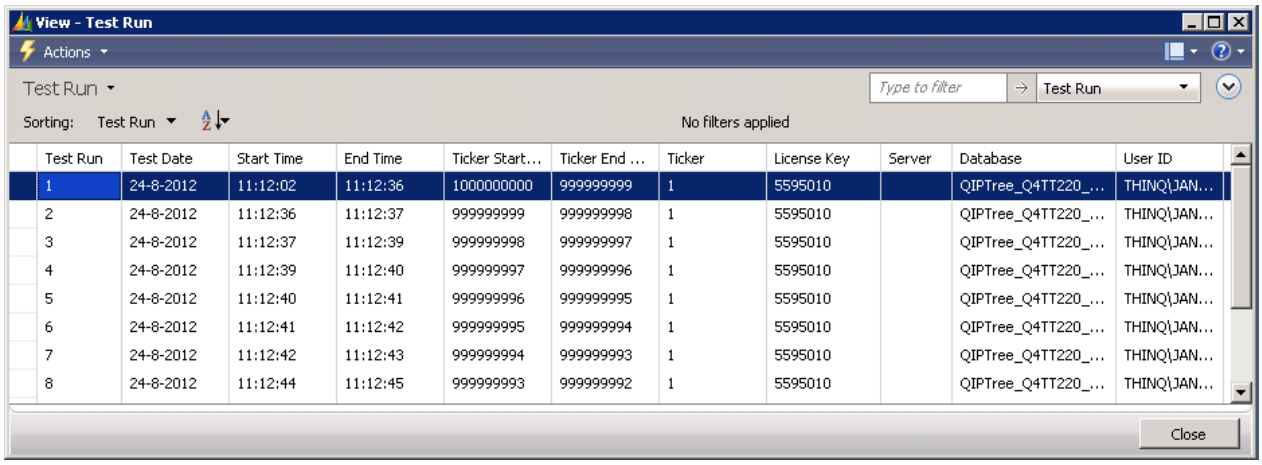


Figure 20: Set Constant with lines filled by Microsoft Excel template

3.3.5 Test run information

When one or more test runs have been executed, information can be found in the test run page.



Test Run	Test Date	Start Time	End Time	Ticker Start...	Ticker End ...	Ticker	License Key	Server	Database	User ID
1	24-8-2012	11:12:02	11:12:36	1000000000	999999999	1	5595010		QIPTree_Q4TT220_...	THINQ\JAN...
2	24-8-2012	11:12:36	11:12:37	999999999	999999998	1	5595010		QIPTree_Q4TT220_...	THINQ\JAN...
3	24-8-2012	11:12:37	11:12:39	999999998	999999997	1	5595010		QIPTree_Q4TT220_...	THINQ\JAN...
4	24-8-2012	11:12:39	11:12:40	999999997	999999996	1	5595010		QIPTree_Q4TT220_...	THINQ\JAN...
5	24-8-2012	11:12:40	11:12:41	999999996	999999995	1	5595010		QIPTree_Q4TT220_...	THINQ\JAN...
6	24-8-2012	11:12:41	11:12:42	999999995	999999994	1	5595010		QIPTree_Q4TT220_...	THINQ\JAN...
7	24-8-2012	11:12:42	11:12:43	999999994	999999993	1	5595010		QIPTree_Q4TT220_...	THINQ\JAN...
8	24-8-2012	11:12:44	11:12:45	999999993	999999992	1	5595010		QIPTree_Q4TT220_...	THINQ\JAN...

Figure 21: Test run overview

The following fields are available

Field	Description
Test run	Unique identification for each test run
Test date	Date on which the test was executed
Start Time	Start time of the test run
End Time	End time of the test run
Ticker Start Position	Start position of the test ticker
Ticker End Position	End position of the test ticker
Ticker	Number of ticks used
License Key	NAV license key number used during the test
Server	SQL server on which the database is running (only available in the NAV classic client)
Database	Database in which the test is performed
UserID	User ID that was logged in during the test

3.4 Expected values testing

In the above paragraphs the creation of a test set is described so a happy flow can be tested. In this paragraph the expected value testing is described

In this manual the receiving of a purchase order is described. This test should be extended with the following tests

Item Ledger Entry

Due to the receiving of goods, the following inventory should be available in warehouse BLUE.

Item	Lot No.	Quantity
1000		10
80216-T	A	10
80216-T	B	5

Vendor Ledger Entry
Due to only receiving the goods and not invoicing them, no vendor ledger entry should be available

Vendor	Amount
10000	<no records>

To create the excepted values, open the test set and select the button expected values

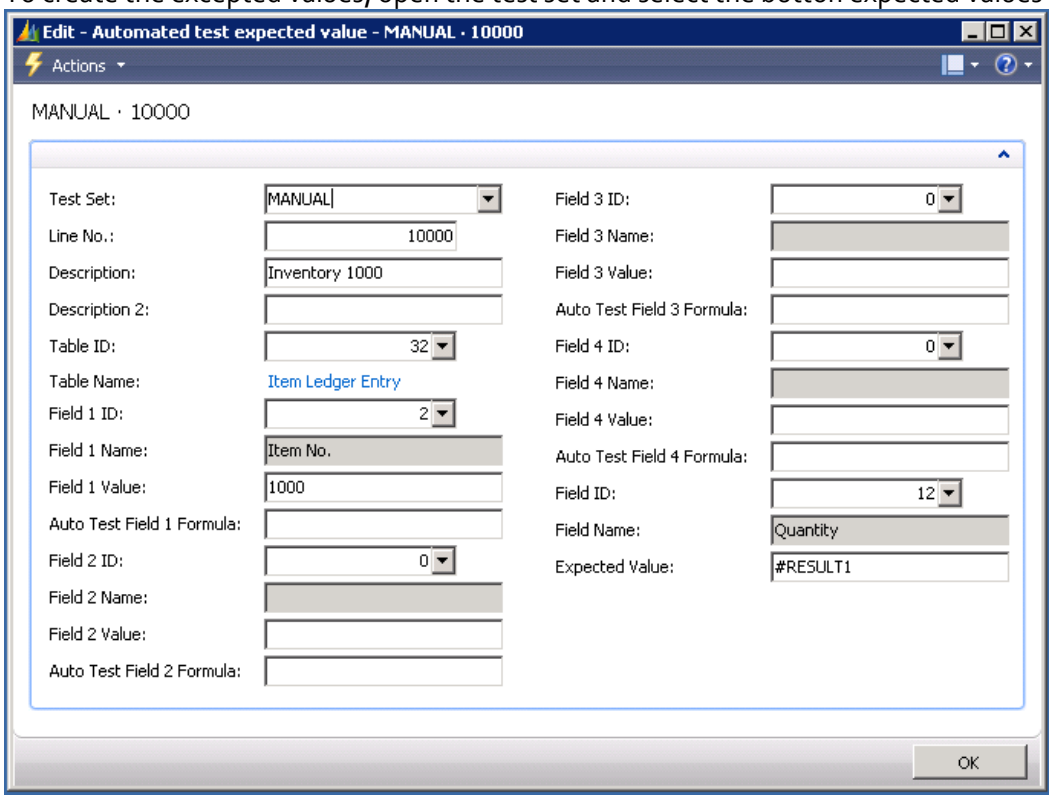


Figure 22: Expected Result card page

Normally a normal value would be expected in the field Expected Value, but two constant sets have been created. The one for the purchase order can be received and the expected value is 10, but a purchase quote can not be received and thus the expected value is 0.

Therefor the constants sets are extended with three new lines to hold the expected values for the tests.

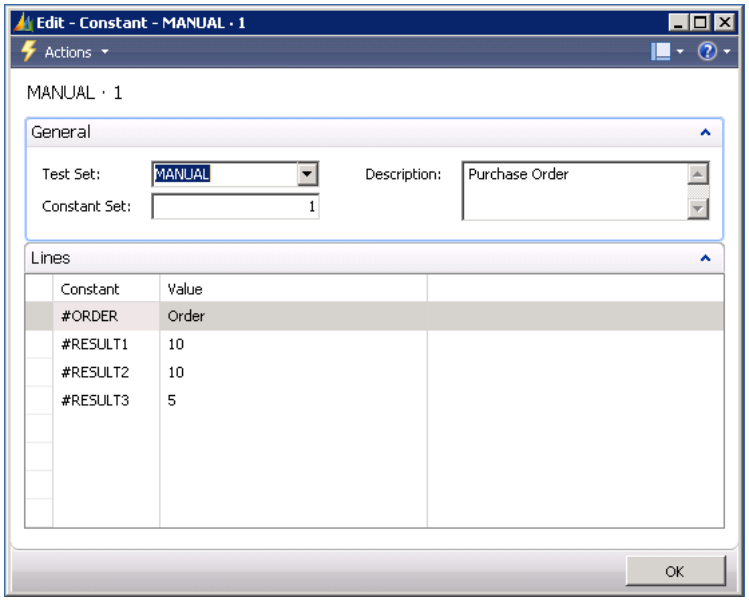


Figure 23: Set Constant card page

The following expected values records are created for the above mentioned test

Item ledger entry test 1

Field	Value
Test Set	Manual
Line No.	10000
Description	Inventory item 1000
Table ID	32
Field 1 ID	2
Field 1 Value	1000
Field ID	12
Expected Value	#RESULT1

Item ledger entry test 2

Field	Value
Test Set	Manual
Line No.	20000
Description	Inventory item 1000
Table ID	32
Field 1 ID	2
Field 1 Value	80216-T
Field 2 ID	6501
Field 2 Value	LOT A

Field ID	12
Expected Value	#RESULT1

Item ledger entry test 3

Field	Value
Test Set	Manual
Line No.	30000
Description	Inventory item 1000
Table ID	32
Field 1 ID	2
Field 1 Value	80216-T
Field 2 ID	6501
Field 2 Value	LOT B
Field ID	12
Expected Value	#RESULT1

Vendor Ledger Entry

Field	Value
Test Set	Manual
Line No.	40000
Description	
Table ID	25
Field 1 ID	1
Field 1 Value	0
Field ID	1
Expected Value	0

4 Creating specific add-on codeunits

To test your own add-on, you have to create at least one codeunit. In this chapter is described how the add-on should be integrated to the QIP4-Testing framework and how this codeunit should be build. This information is use full in case you have bought the QIP4-Testing Development license and only for NAV developers, there is no functional information described in this chapter.

There are two ways to connect to the QIP4-Testing Framework:

- Manually, described in paragraph 4.2
- Automatically, described in paragraph 4.3

4.1 Integration to the QIP4-Testing Framework

To integrate the add-on codeunit into the QIP4-Testing Framework the codeunit 11022366 TT-Development must be modifield.

4.1.1 Vendor information

For each function the vendor is given, so in case the function is not working as expected, the responsible vendor can be contacted.
In the function FillVendor(), table the information for QIPTree is already provided. Please add a Vendor record with your information.

```
CLEAR(Vend);
Vend.Code := '<Code>';
Vend.Name := '<Name>';
Vend.Address := 'Address';
Vend."Post Code" := '<Post Code>';
Vend.City := '<City>';
Vend.Contact := '<Support email address>';
Vend."Phone No." := '<Support telephone number>';
Vend.INSERT;
```

4.1.2 Codeunit information

To initialize, to execute the add-on specific function and to delete the add-on specific historical data, the codeunit itself must be initialized. This is done via adding it to the initialization function FillVendorCodeunit() information.
Please use as free number an integer between 50 and 100. The Vendor code should be the same as specified in paragraph 4.1.1.

```
11:
  BEGIN
    VendCdu.Vendor := 'QIPTree';
    VendCdu."Codeunit ID" := 11210246;
    VendCdu.Description := 'QIP4-Inventory Status 2009 Functions';
  END;

     <Free number>:
  BEGIN
    VendCdu.Vendor := '<Code>';
    VendCdu."Codeunit ID" := <Codeunit ID>;
```

```
      VendCdu.Description := '<Codeunit description';
    END;
  END;
  IF VendCdu."Codeunit ID" <> 0 THEN BEGIN
    VendCdu.INSERT;
  END;
END;
```

4.2 Add-on integration Codeunit

The add-on integration codeunit should have a couple of standard functions. Please keep in mind that code in this codeunit should be kept as simple as possible. This codeunit is intended to execute add-on specific functions, not to make complex coding.

4.2.1 OnRun

In the QIP4-Testing framework a call can be made to available functions in the function library. Only the codeunits that are mentioned in the Vendor Codeunit table will be accessed.

```
CodeunitID := <Codeunit ID>;

FunctionType := GlobInfo.GetFunctionType;

CASE FunctionType OF
  FunctionType::" ", FunctionType::Initialize: InitFunct();
  FunctionType::Execute: ExecFunct();
  FunctionType::Delete: DeleteProcesData();
END;

GlobInfo.SetFunctionType(FunctionType::" ");
```

4.2.2 Initialize add-on functions

As mentioned in paragraph 4.2.1. the function InitFunct() will be called when the QIP4-Testing framework wants to initialize all functions that it can use.
The function has two main parts, one to identify the function and one to add the function parameters. As an example the function "Receive purchase order" will be used.

The first identifies the function, so it should have an unique number, code and description

```
Function: InitFunct()

FunctionSetup.RESET;
FunctionSetup.SETRANGE("Codeunit ID",CodeunitID);
FunctionSetup.DELETEALL(TRUE);

VendorCodeunit.SETRANGE("Codeunit ID",CodeunitID);
Ok := VendorCodeunit.FINDFIRST;

FOR i := 1 TO 200 DO BEGIN
  CLEAR(FunctionSetup);
```

```
CASE i OF
        // Purchase
        1: FunctionSetup.Code := 'RELEASE PO';
        2: FunctionSetup.Code := 'REOPEN PO';
        3: FunctionSetup.Code := 'RECEIVE PO';
        ...
        131: FunctionSetup.Code := 'GET STD. SALES CODES';  //
END;
CASE i OF
        1: FunctionSetup.Description := 'Release Purchase Order';
        2: FunctionSetup.Description := 'Reopen Purchase Order';
        3: FunctionSetup.Description := 'Receive Purchase Order';

END;
FunctionSetup.Vendor := VendorCodeunit.Vendor;
FunctionSetup."Codeunit ID" := CodeunitID;
FunctionSetup."Execution Number" := i;
IF FunctionSetup.Code <> '' THEN BEGIN
        FunctionSetup.INSERT(TRUE);
END;
```

In the second part you should define which parameters should be added to the function. For the example function "Receive purchase order" two parameters are needed, namely the document type and document number. By using these two parameters the correct purchase header can be found and the function can be executed.

```
/// ------------------------------------------------------------
/// Function Parameters
/// ------------------------------------------------------------
CASE i OF
 1,2,3,4,5,6,7:
        FOR j := 1 TO 2 DO BEGIN
          CLEAR(FunctionParameters);
          FunctionParameters.Vendor := VendorCodeunit.Vendor;
          FunctionParameters.Code := FunctionSetup.Code;
          FunctionParameters."Line No." := j * 10000;
          CASE j OF
            1: FunctionParameters.Description := 'Document type';
            2: FunctionParameters.Description := 'Document number';
          END;
          FunctionParameters.INSERT(TRUE);
        END;
END;
```

When you have made the changes, press the initialize button and then the function should be available.

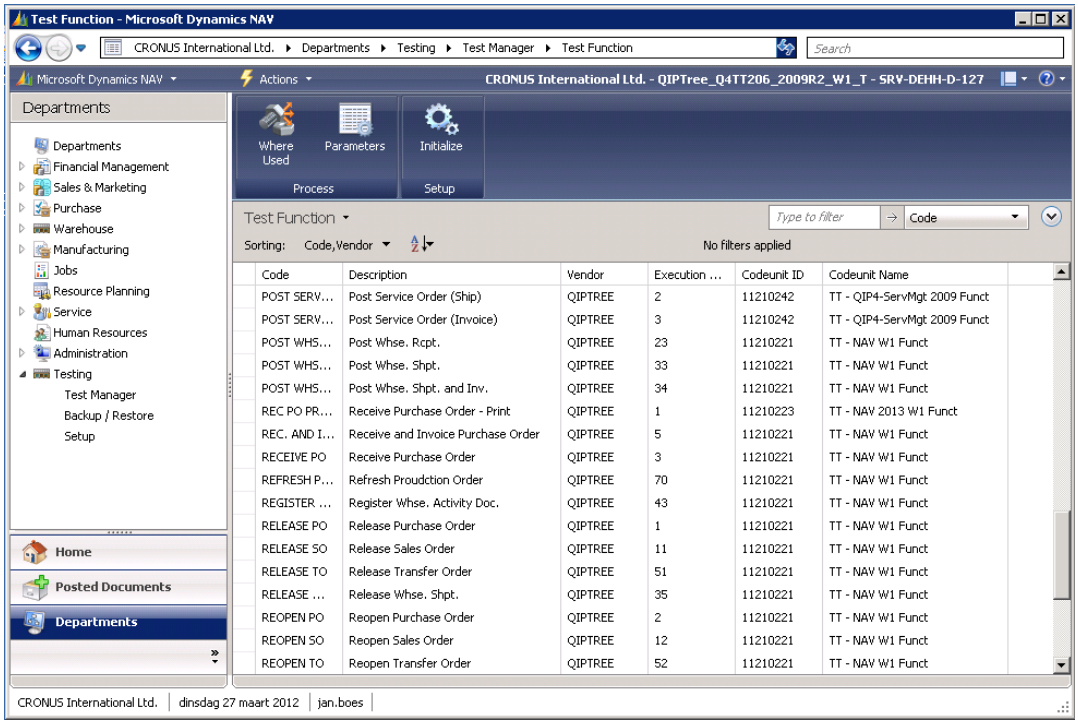


Figure 24: Overview of available test functions (including the new ones)

4.2.3 Execute functions

As shown in the figure above, the function Receive PO is available in codeunit 11210221 and has execution ID number 3.
This number is the starting point in function ExecFunct().

```
Function: ExecFunct()
FunctionSetup.GET(GlobInfo.GetFunctionCode(),GlobInfo.GetVendor());
UIMgt.GetFunction(SetFunct,GlobInfo.GetFunctionCode(),GlobInfo.GetFunctionLine());

CASE FunctionSetup."Execution Number" OF
  1: ReleasePurchHeader();
  2: ReopenPurchHeader();
  3: ReceivePurcHeader();
END;
```

So execution ID 3 is found and the real integration function is started. Here is the code:

```
Function: ReceivePurcHeader()

GlobInfo.SetStrMenuChoice(1);    //Receive

UIMgt.GetFunctionLine(SetFunctLine,FunctionSetup.Code,10000);  //Doc Type
UIMgt.ConvertValue(SetFunctLine.Value,'');
Value1 := SetFunctLine.Value;
EVALUATE(i,Value1);

UIMgt.GetFunctionLine(SetFunctLine,FunctionSetup.Code,20000);  //Doc No.
```

```
UIMgt.ConvertValue(SetFunctLine.Value,'');
Value2 := SetFunctLine.Value;

PurchHeader.GET(i,Value2);

IF SetFunct."Error Cause" THEN BEGIN
  ASSERTERROR PurchPostYN.RUN(PurchHeader);
  UIMgt.WriteErrorMessage();
END ELSE BEGIN
  PurchPostYN.RUN(PurchHeader);
END;
```

I think that some explanation is in place

In the codeunit TT – Global information all kind of global functions have been defined. The function SetStrMenuChoice means that the strmenu handler is set to 1.

```
GlobInfo.SetStrMenuChoice(1);    //Receive
```

In the initialization part we have added two parameters, one for the document type and one for the document number. The information is retrieved from the databases and used to get the correct purchase header.

```
UIMgt.GetFunctionLine(SetFunctLine,FunctionSetup.Code,10000);  //Doc Type
UIMgt.ConvertValue(SetFunctLine.Value,'');
Value1 := SetFunctLine.Value;
EVALUATE(i,Value1);

UIMgt.GetFunctionLine(SetFunctLine,FunctionSetup.Code,20000);  //Doc No.
UIMgt.ConvertValue(SetFunctLine.Value,'');
Value2 := SetFunctLine.Value;

PurchHeader.GET(i,Value2);
```

In the test set function card you can determine if the function will cause an error (in this example if you want to receive an order with quantity 0) and then the ASSERTERROR will surpress the error message that will come in the add-on function. Otherwise the add-on function will be executed.

```
IF SetFunct."Error Cause" THEN BEGIN
  ASSERTERROR PurchPostYN.RUN(PurchHeader);
  UIMgt.WriteErrorMessage();
END ELSE BEGIN
  PurchPostYN.RUN(PurchHeader);
END;
```

4.2.4 Delete add-on historical data

In paragraph 4.1.2 we have created the vendor codeunit. The QIP4-Testing framework will run through all listed codeunits and when the object is available it will execute the function DeleteProcesData(). In this function all tables should be listed for which the data should be removed so a regression test (every time the same result) can be achieved.

```
DeleteProcesData()
```

T17.DELETEALL;
T21.DELETEALL;
T25.DELETEALL;
T32.DELETEALL;
T36.DELETEALL;
T37.DELETEALL;
T38.DELETEALL;
....

4.2.5 Global Variables

The following global variables should be available in the codeunit

Name	DataType	Subtype	Length	Option Values
FunctionSetup	Record	TT - Function Setup		
SetFunct	Record	TT - Set Function		
SetFunctLine	Record	TT - Set Function Line		
VendorCodeunit	Record	TT - Vendor Codeunit		
GlobInfo	Codeunit	TT - Global Information		
UIMgt	Codeunit	TT - UI Mgt		
Value1	Text		254	
Value2	Text		254	
Value3	Text		254	
FunctionType	Option			,Initialize,Execute,Delete,MasterData
CodeunitID	Integer			

4.3 Function Generator

This function generator will create the above mentioned code automatically. Of course you have to test if these functions are working correct before adding them to your test scripts, but the development time needed to create the code will be decreased significantly.

The following steps should be executed to create the integration codeunit.

1. Create a Microsoft Dynamics NAV text file from the add-on codeunit.
2. Start the function Generate Functions in the Function Generation page
3. Modify function code and - names and loop values. These will be used as function code and nema when the new object file is created.
4. Start the function Create NAV object file in the Function Generation page
5. Import the created Microsoft Dynamics NAV text file and compile the object

The Function Generator page can be found in Department menu – Testing – Function Generator – Function Generator.

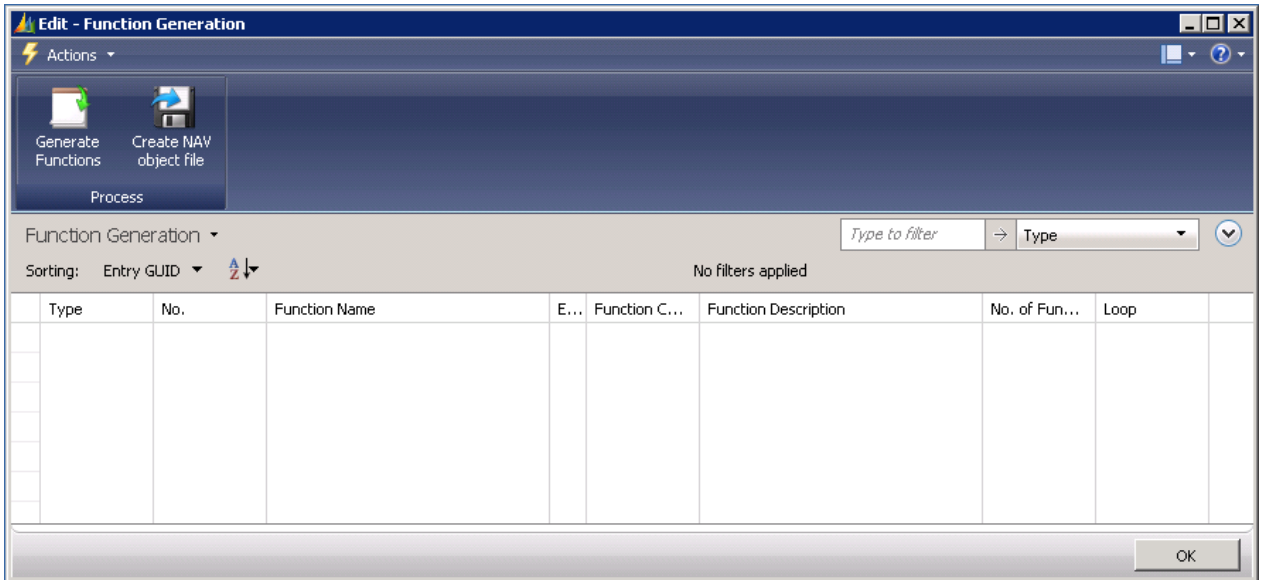


Figure 25: Function Generation list

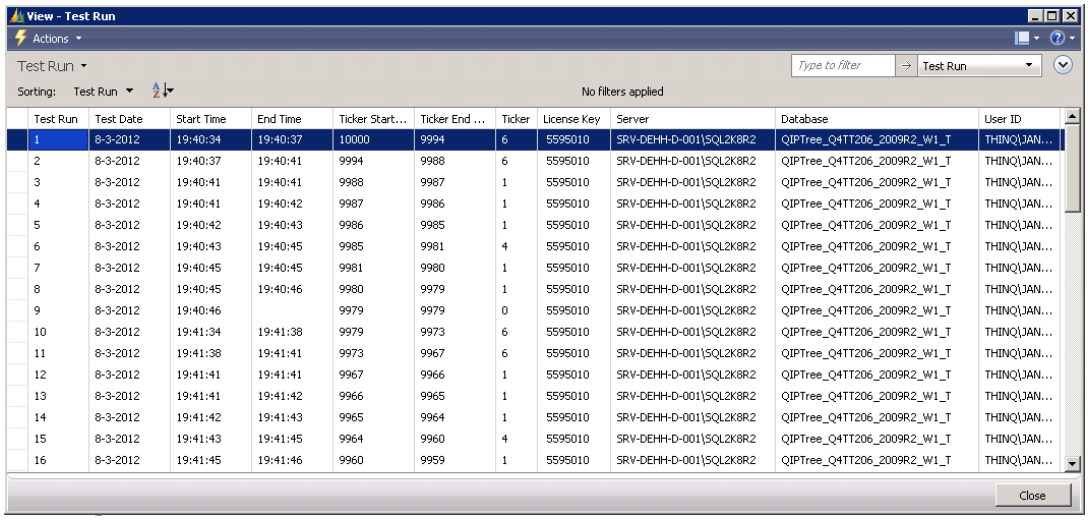
The following fields are available

Field	Description
Type	NAV object type
No.	NAV object number
Function Name	NAV function name
Execute	In case this field is activated, a function will be created in het NAV object file
Function Code	The function code in the QIP4-Testing function library that will identify the NAV function
Function Description	The function description in the QIP4-Testing function library that describes the NAV function
No. of Function	Default 1, needed as a counter
Loop	In some cases a popup is used in the codeunit to give the user the possibility to select an option (like posting a purchase order, do you want to receive, invoice, receive and invoice). In this case the loop should be 3. Internally the system will create the code which is needed to automatically select this option.

5 Cloud Connector

The licensing model for QIP4-Testing consists of two parts: a granule price and a pay-by-use part. For every test run that will be executed a specific price will be charged. To keep this process simple QIPtree will send an invoice and when paid the available number of test runs will be increased. The administration application is created as a cloud application and every time a test run will be executed, it will check against the administration application to see if it is allowed to be executed.

In the NAV application we have created an test run overview to get information who executed one or more test runs. The page can be found via Departments > Testing > Test Manager > Test Run



Test Run	Test Date	Start Time	End Time	Ticker Start...	Ticker End ...	Ticker	License Key	Server	Database	User ID
1	8-3-2012	19:40:34	19:40:37	10000	9994	6	5595010	SRV-DEHH-D-001\SQL2K8R2	QIPTree_Q4TT206_2009R2_W1_T	THINQ\JAN...
2	8-3-2012	19:40:37	19:40:41	9994	9988	6	5595010	SRV-DEHH-D-001\SQL2K8R2	QIPTree_Q4TT206_2009R2_W1_T	THINQ\JAN...
3	8-3-2012	19:40:41	19:40:41	9988	9987	1	5595010	SRV-DEHH-D-001\SQL2K8R2	QIPTree_Q4TT206_2009R2_W1_T	THINQ\JAN...
4	8-3-2012	19:40:41	19:40:42	9987	9986	1	5595010	SRV-DEHH-D-001\SQL2K8R2	QIPTree_Q4TT206_2009R2_W1_T	THINQ\JAN...
5	8-3-2012	19:40:42	19:40:43	9986	9985	1	5595010	SRV-DEHH-D-001\SQL2K8R2	QIPTree_Q4TT206_2009R2_W1_T	THINQ\JAN...
6	8-3-2012	19:40:43	19:40:45	9985	9981	4	5595010	SRV-DEHH-D-001\SQL2K8R2	QIPTree_Q4TT206_2009R2_W1_T	THINQ\JAN...
7	8-3-2012	19:40:45	19:40:45	9981	9980	1	5595010	SRV-DEHH-D-001\SQL2K8R2	QIPTree_Q4TT206_2009R2_W1_T	THINQ\JAN...
8	8-3-2012	19:40:45	19:40:46	9980	9979	1	5595010	SRV-DEHH-D-001\SQL2K8R2	QIPTree_Q4TT206_2009R2_W1_T	THINQ\JAN...
9	8-3-2012	19:40:46		9979	9979	0	5595010	SRV-DEHH-D-001\SQL2K8R2	QIPTree_Q4TT206_2009R2_W1_T	THINQ\JAN...
10	8-3-2012	19:41:34	19:41:38	9979	9973	6	5595010	SRV-DEHH-D-001\SQL2K8R2	QIPTree_Q4TT206_2009R2_W1_T	THINQ\JAN...
11	8-3-2012	19:41:38	19:41:41	9973	9967	6	5595010	SRV-DEHH-D-001\SQL2K8R2	QIPTree_Q4TT206_2009R2_W1_T	THINQ\JAN...
12	8-3-2012	19:41:41	19:41:41	9967	9966	1	5595010	SRV-DEHH-D-001\SQL2K8R2	QIPTree_Q4TT206_2009R2_W1_T	THINQ\JAN...
13	8-3-2012	19:41:41	19:41:42	9966	9965	1	5595010	SRV-DEHH-D-001\SQL2K8R2	QIPTree_Q4TT206_2009R2_W1_T	THINQ\JAN...
14	8-3-2012	19:41:42	19:41:43	9965	9964	1	5595010	SRV-DEHH-D-001\SQL2K8R2	QIPTree_Q4TT206_2009R2_W1_T	THINQ\JAN...
15	8-3-2012	19:41:43	19:41:45	9964	9960	4	5595010	SRV-DEHH-D-001\SQL2K8R2	QIPTree_Q4TT206_2009R2_W1_T	THINQ\JAN...
16	8-3-2012	19:41:45	19:41:46	9960	9959	1	5595010	SRV-DEHH-D-001\SQL2K8R2	QIPTree_Q4TT206_2009R2_W1_T	THINQ\JAN...

Figure 26: Overview of executed test runs

Field	Description
Test Run	Unique number created for every test run
Test Date	Date when the test run was executed
Start Time	Start time when the test run was executed
End Time	End time when the test run was finished
Ticker Start position	Start position of the test ticker
Ticker End position	Available test run after execution of the test runs
Ticker	Number of executed test runs
License Key	Microsoft Dynamics NAV License Key
Server	SQL server database on which the database is running that is used by QIP4-Testing (field is only filled when QIP4-Testing is running via the Classic client).
Database	Microsoft Dynamics NAV database that is used by QIP4-Testing
User ID	User ID who executed the test runs.

To make a connection to the cloud application the following information should be filled in the QIP4-Testing setup page. The page can be found via Departments > Testing > Setup > Setup

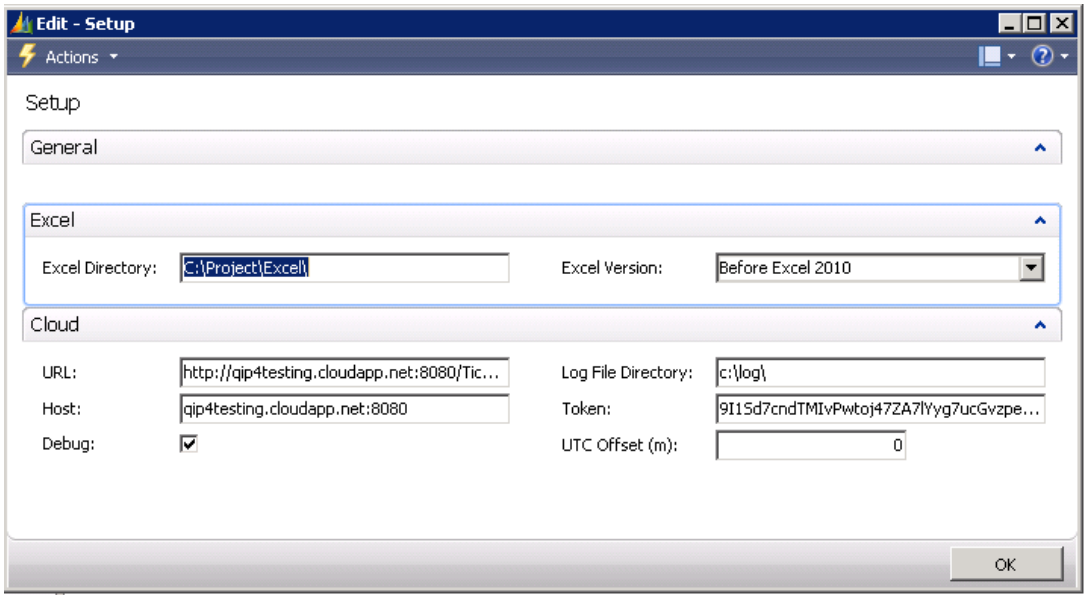


Figure 27: QIP4-Testing setup

Field	Description
Excel Directory	Directory where excel files are placed
Excel version	Due to changes in the file structure (.xls vs. .xlsx) the appropriate Microsoft Office Excel version has to be selected that is installed on the client. In case the option "Before Excel 2010" is selected a .xls file will be created. In case the option "Excel 2010" is selected a .xlsx file will be created
URL	URL to connect to the cloud application
Host	Host to connect to the cloud application
Debug	In case no connection can be made or an error message appears, the XML files will be copied to the specified directory in case the field "Debug" is activated.
Log File Directory	Directory where the debug information is stored.
Token	Secure connection string to connect to the cloud application
UTC Offset	The UTC offset (offset from Coordinated Universal Time) is the difference in hours and minutes from Coordinated Universal Time (UTC) for a particular time.

The correct URL, Host and token information will be provided when QIP4-Testing Cloud Connector is bought.

6 Walk through – Creating a test set

Before you can start building the test set, you first have to define what should be tested. In this example a purchase order must be received but not invoiced.

The purchase order must be created with the following information.

Purchase order header:

Field	Values
Buy-from vendor No.	10000
Location Code	BLUE

Purchase order line (1):

Field	Values
Type	Item
No.	1000
Quantity	10
Unit of Measure Code	PCS
Direct Unit Cost	1250

Purchase order line (2):

Field	Values
Type	Item
No.	80216-T
Quantity	15
Unit of Measure Code	PCS
Direct Unit Cost	0,50

Purchase order line (2), lot tracking information (1):

Field	Values
Lot No.	A
Quantity	10

Purchase order line (2), lot tracking information (2):

Field	Values
Lot No.	B
Quantity	5

6.1.1 Test set definition

Based on the above mentioned information, the following test set has to be created

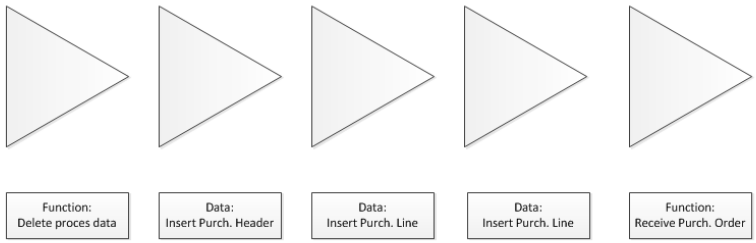


Figure 28: Test steps to be taken to create a test set

The test set has the following parts:

- Delete proces data: to remove old test data
- Insert purch. header: to create a purchase order header
- Insert purch. line: to create a purchase order line
- Insert purch. line: to create a purchase order line
- Receive Purch. order: to receive to created purchase order

6.1.2 Test set

The test set consist of a header (for identification) and the test lines which contains the real test script.

Set header

Field	Values
Test Set	MANUAL
Description	Walk through receiving a purchase order
Description 2	
Status	Open
Error Found	FALSE

To add the needed functions you need to select the set function called "Functions". In the list page you have to create the following two lines with information

Set function "Delete process data"

Field	Values
Test Set	MANUAL
Code	DELETE PROCES DATA
Description	Delete process data
Parameters	FALSE

Set function "Receive Purchase Order"

Field	Values
Test Set	MANUAL
Code	RECEIVE PD
Description	Receive Purchase Document
Parameters	TRUE

Now you have to create the link between the functions and the purchase order that should be received. This is done via the function parameters

Field	Values
Line No.	10000
Description	Document type
Value	@PORDER_TYPE

Field	Values
Line No.	20000
Description	Document no.
Value	@PORDER_NO

6.1.3 Create Data blocks for the test set

Purchase Order Header

The first data block must create the purchase order header.

Data block header

Field	Values
Element	MANUAL 01
Description	Purchase Order – 10000 – BLUE
Table	38
Key No.	1
Key Action	Insert

Data block lines

Type	Field ID	Field Name	Value
Group		Key	
Key	1	Document Type	Order
Key	3	No.	
Group		Field	
Field	2	Buy-from Vendor No.	10000
Field	28	Location Code	BLUE

Group		Variable	
Variable	1	Document Type	@PORDER_TYPE
Variable	3	No.	@PORDER_NO

Purchase Order Line 1

The second data block must create the first purchase order line.

Data block header

Field	Values
Element	MANUAL 01 – 1
Description	Purchase Order Line – 1000 – 10 PCS
Table	39
Key No.	1
Key Action	Insert

Data block lines

Type	Field ID	Field Name	Value
Group		Key	
Key	1	Document Type	@PORDER_TYPE
Key	3	Document No.	@PORDER_NO
Key	4	Line No.	
Group		Field	
Field	5	Type	Item
Field	6	No.	1000
Field	15	Quantity	10
Field	5407	Unit of Measure Code	PCS
Field	22	Direct Unit Cost	1250
Group		Variable	
Variable	4	Line No.	@PORDER_LINE

Purchase Order Line 2

The third data block must create the second purchase order line.

Data block header

Field	Values
Element	MANUAL 01– 2
Description	Purchase Order Line – 80216-T – 15 PCS
Table	39
Key No.	1
Key Action	Insert

Data block lines

Type	Field ID	Field Name	Value
Group		Key	
Key	1	Document Type	@PORDER_TYPE
Key	3	Document No.	@PORDER_NO
Key	4	Line No.	
Group		Field	
Field	5	Type	Item
Field	6	No.	80216-T
Field	15	Quantity	15
Field	5407	Unit of Measure Code	PCS
Group		Variable	
Variable	4	Line No.	@PORDER_LINE
Variable	10	Expected Receipt Date	@EXP_RCPT_DATE

Reservation Entries

Now the reservation entries have to be created. Creating them via the standard NAV functionality is not possible, because the creating is done via a page and that can be tested. So the end result has to be created directly.

To determine the next reservation entry number we have to define a data block that looks for the last reservation entry and add 1 to get the next reservation number.

Data block header

Field	Values
Element	GET_NEXT_RESERVENTRY
Description	Get next reservation entry
Table	337
Key No.	1
Key Action	FindLast

Data block lines

Type	Field ID	Field Name	Value	Formula
Group		Key		
Key	1	Entry No.		
Key	28	Positive	TRUE	
Group		Field		
Group		Variable		
Variable	1	Entry No.	@RESERV_ENTRY	+1

Now all information is acquired and the reservation entries can be created. Because the purchase line has two lots, two reservation entries have to be created.

Lot A

Data block header

Field	Values
Element	MANUAL 01– 2-1
Description	80216-T – LOT A - 10 PCS
Table	337
Key No.	1
Key Action	Insert

Data block lines

Type	Field ID	Field Name	Value
Group		Key	
Key	1		@RESERV_ENTRY
Key		Document No.	TRUE
Key	4	Line No.	
Group		Field	
Field	3	Location Code	BLUE
Field	29	Qty. per Unit of Measure	1
Field	4	Quantity (Base)	10
Field	5	Reservation Status	Surplus
Field	8	Creation Date	WORKDATE
Field	10	Source Type	39
Field	11	Source Subtype	@PORDER_TYPE
Field	12	Source ID	@PORDER_NO
Field	15	Source Ref. No.	@PORDER_LINE
Field	22	Expected Receipt Date	@EXP_RCPT_DATE
Field	5400	Lot No.	A
Field	6510	Item Tracking	Lot No.
Group		Variable	

Lot B

Data block header

Field	Values
Element	MANUAL 01-2-2
Description	80216-T – LOT B - 5 PCS
Table	337
Key No.	1

Key Action	Insert

Data block lines

Type	Field ID	Field Name	Value
Group		Key	
Key	1		@RESERV_ENTRY
Key		Document No.	TRUE
Key	4	Line No.	
Group		Field	
Field	3	Location Code	BLUE
Field	29	Qty. per Unit of Measure	1
Field	4	Quantity (Base)	5
Field	5	Reservation Status	Surplus
Field	8	Creation Date	WORKDATE
Field	10	Source Type	39
Field	11	Source Subtype	@PORDER_TYPE
Field	12	Source ID	@PORDER_NO
Field	15	Source Ref. No.	@PORDER_LINE
Field	22	Expected Receipt Date	@EXP_RCPT_DATE
Field	5400	Lot No.	B
Field	6510	Item Tracking	Lot No.
Group		Variable	

6.1.4 Put it all together in one test set

The functions and data blocks are created and the test set can be completed by adding the lines.

Set lines

Execute	Type	Code	Description
TRUE	Function	DELETE PROCES DATA	Delete process data
TRUE	Data	MANUAL 01	Purchase Order – 10000 – BLUE
TRUE	Data	MANUAL 01-1	Purchase Order Line – 1000 – 10 PCS
TRUE	Data	MANUAL 01-2	Purchase Order Line – 80216-T – 15 PCS
TRUE	Data	GET_NEXT_RESERVENTRY	Get next reservation entry
TRUE	Data	MANUAL 01-2-1	80216-T – LOT A - 10 PCS
TRUE	Data	GET_NEXT_RESERVENTRY	Get next reservation entry
TRUE	Data	MANUAL 01-2-2	80216-T - LOT B - 5 PCS
TRUE	Function	REVEICE PD	Receive Purchase Document