



ZTEST User documentation





ZTEST User documentation

Contents

Changes in release V2.001	6
Changes in release V1.001	6
Original release and enhancements.....	6
Document update 2020-06-11	6
Document update 2006-06-18.....	7
The section on how to install the product has been updated.....	7
Document and functional update 2006-06-23.....	7
What is in the package?	7
More in detail:.....	7
Licensing.....	9
Disclaimer	9
Download and setup	10
URL for download.....	10
Extract the contents of the Zip file.....	10
Prepare for install on your system	11
Install product	11
TNAPI library.....	12
ZTEST menu	13
Functions for scripted testing	14
ZTSTGEN – Test data generation.....	15
What is this used for?.....	15
What it does:	15
What is required:.....	15
How to use the function.....	16
Details per field	19
When all criteria have been entered	22
Creating the program	23
If there are errors?	23



ZTEST User documentation

What the code looks like	24
Declarations.....	24
Initial processing.....	25
Main processing	25
Finally	25
Resulting data.....	26
What if I have special requirements for fields – such as external procedures?	26
If you do manual changes to the generated program	27
Other potential issues	27
Final tip.....	27
ZCMPGEN – Compare versions of files.....	27
What is this used for?.....	27
What it does:	27
Merge file	27
“Miss files”	28
Difference file	28
What the function does not do:.....	28
What is the performance impact of using the tool?	28
How to use the function.....	29
First.....	29
Main screen.....	29
Select output files and source	30
Example:	31
Select unique access path	32
Select matching key fields	33
When you have defined your key fields.....	33
Select fields to check for differences	34
Selecting another field from “new” file (option 2)	35
Generate source and compile	36
What has been created?	37
MRG0528 a view that combines ALL fields from both files where the keys match.	37



ZTEST User documentation

MNE0528 records in OLD missing in NEW	37
MOL0528 records in NEW missing in OLD	37
MDI0528 records with differences in selected new key fields	37
Scripted testing.....	38
What this does.....	38
How this works	38
Required setup	39
Recording a macro.....	40
Stop recording macro	41
Transfer macro(-s) to IFS	41
Convert recorded macro to script file	41
Why is this step necessary?.....	41
What happens?	41
Working with scripts.....	43
Available options for the scripts.....	44
2 – Edit a script	44
3 – Copy a script	44
4 – Delete a script.....	44
5 – View a script	44
S – Submit a script	44
D – View database changes.....	44
V – View screen data	46
Function key F6 - Converting recorded macros to scripts	46
How TNAPI scripts are formatted	48
DATA Statement	48
SEND Statement	48
Available cursor control codes for the “DATA” statement	48
Available control codes for the SEND statement.....	49
This shows you	52
Updating the package	53
Finding the updates.....	53



ZTEST User documentation



ZTEST User documentation

ZTEST package

Changes in release V2.001

The package runs on release V7R3M0 and later releases.

All objects in save file are now owned by user QPGMR.

- All requirements for a license key are now removed. As is the menu option to enter said key. The package is now provided free of charge and with the source code provided.
- Changes in maximum sizes for macro conversion.
- Clarifications for file compare generation (display)
- All source code included in package.

Changes in release V1.001

- Test data generator:
 - Changes in format of generated program
 - Data now written from qualified data structure (handles case when same field name exists in target and retrieved files)
 - More readable format of generated code
 - Retrieved files and fields are now validated online.
 - Formula field is formatted after entry.
- Scripted tests
 - Macro to Script conversion is amended and re-instated.

Original release and enhancements

Is date 2020-06-10 and has version number V1.000

Enhancements for further releases and update packages will be listed in future versions of this document.

Document update 2020-06-11

In the first version of this document a portion of the documentation on script handling was missing.

This has now been included. No changes in the software has occurred. This version of the document is included in the download package and is also available for separate download on the PTF page.



ZTEST User documentation

Document update 2006-06-18

The section on how to install the product has been updated.

Document and functional update 2006-06-23

Some amendments to functions for file compare and test data generation have been introduced.

Also, the function to convert ACS macros to TNAPI scripts has been temporarily disabled.

This function will be restored in V1.001 (full product load and PTF package) that will be released later this summer. The reason for this are some issues with the encoding of the ACS macro data when retrieved.

What is in the package?

This software package consists of Three functions designed to make it easier to test software changes:

1. A test data generator that creates RPG programs that generate data in files based on rules that you specify. The data is created by a randomization process, and you can have as many records that you like created for you. And what you get will look and feel like actual data.
2. A set of functions for scripted testing of online functions. This enables you to record and save keystrokes as a script-macro file and submit it for execution in batch. All displayed screens are saved for later analysis. As will the database changes that occurred when the script was executed.
3. Functions to create views for comparison of data in two files. Thus, is frequently used for the purpose of verifying that old and new program versions create the same data.

One view will always contain all old and new file fields joined by primary keys that you provide. Other optional views show records that exist in one file but is missing in the other. Lastly one view show records where keys match but specified fields have different values.

Although all three functions are called as commands there is a menu available which also contains management functions for application settings.

More in detail:

- Programs (RPG and CL) for:
 - Setup and admin purpose (such as folders for the run scripts function)
 - File compare functions.
 - Macro/Script handling, conversion, running checking results.
 - Test data generation
- One menu **ZTEST** with access to all functions in the package



ZTEST User documentation

- Source files (suggested only – you may use other locations and file names):
 - QFLDSRC – this is used (suggested) for your field level source statements for test data generation.
 - QTNSCRIPT – for storing scripts for running scripted tests.
 - ZCOPYSRC – has control entries for created test data generator programs.
- Database sample files for file compare function.
 - SAMPLEOLD “order file” with two records with matching keys for “new” file one of which has data fields that differ. The other with same data. One record which does not exist in “new” file.
 - SAMPLENEW “order” file with one record that does not exist in “old” file.
- Display files for all functions
- Three commands:
 - ZCMPGEN – runs function to create SQL views to compare datasets.
 - ZRUNTNSCR – submits a test script.
 - ZTSTGEN – runs function to create a test data generator program.
- Two data areas
 - ZADMIN settings for folders (PC and IFS) for macros, journaling status and more specifically for running scripted testing.
 - ZCMPGEN – saves locations for generated source and objects.
- One binding directory ZTDGEN this is used when compiling test data generator programs.
- One module ZRANDOM this has all the procedures needed by the created test data generator program.



ZTEST User documentation

Licensing

No, the product is provided totally free of charge for unlimited use. If you require personal assistance, I will charge you for that as per standard pricelist.

But I would appreciate any suggestions for improvements or changes so that we together can make the package even better.

Disclaimer

You take the responsibility for any consequences of using the product. It is all up to you – not to me!

Fair don't you think?

Also: Now that the source code is included I – at a time – had the idea to “beautify” the code with copious more comments and such. As I have not quite had the time to do this I include the running code as is. I have instead perform a lot of testing to verify that it all works as designed.

(Not that for scripted testing you WILL need to download and install the TNAPI software from another source!)



ZTEST User documentation

Download and setup.

URL for download

Navigate to this URL:

<http://konsult.karlpersaker.se/downloads/ztest-package/>

And download the Zip file that contains the user documentation and the save file with the program product.

Download

Download this zip file and extract it to your PC.

Then follow the instructions in the user documentation.



 REDIGERA

Click on the green button and save the file on your system.

Extract the contents of the Zip file.

The Zip file contains two objects:

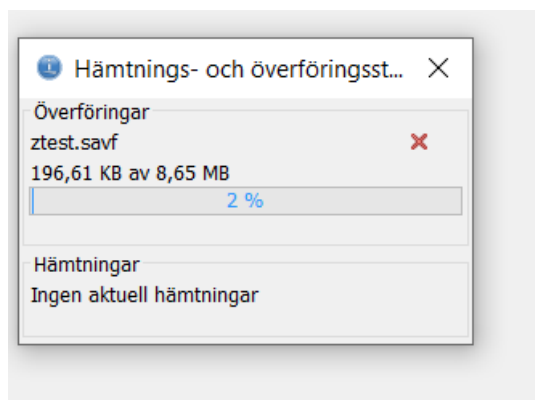
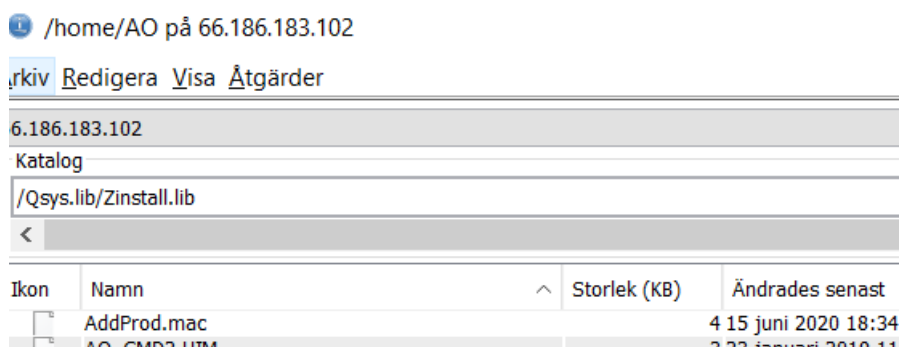
- The full user documentation (this document although the install section is available as a separately downloadable document for convenience purposes)
- A save file from the IBM I system named ZTEST.SAVF. This contains the entire program package with all required objects.



ZTEST User documentation

Prepare for install on your system.

- Easiest way:
 - Use the “Integrated file system” function in ACS.
 - Navigate to the root and to QSYS.LIB.
 - Navigate from there to a library on you IBM i where you can place your save file.
 - Use Ctrl + U to transfer the file to your IBM i library of choice.
 - Restore the library ZTEST from the save file.



- FTP Option:
 - Create a save file in a library on your system.
 - Use FTP (binary transfer) to “put” the .savf file to the library where you created the save file.
 - Restore the library ZTEST from the save file.

Install product.

When the ZTEST library is restored from the save file the owner is set to QPGMR.



ZTEST User documentation

TNAPI library

The TNAPI library is a separate download and install from the website of the developer (the late Albert York).

TNAPI is the engine behind running the 5250 scripts in a batch environment.


The piece that is in the ZTEST library deal with conversion of recorded macros and managing the results of the submitted runs.

And – yes: A portion of the proceeds for ZTEST are forwarded to Albert York!

You find this download here:



TNAPI - An API that allows batch programs to run interactive commands and programs and RUNTNSCR a utility that runs scripts using TNAPI

 TNAPI.zip
Size : 54.035 Kb
Type : zip

This is the file to download and save.

Expand this on your desktop system.

Then follow the instructions in the .TXT document.



ZTEST User documentation

Application menu

ZTEST menu

Go to the application menu (GO ZTEST)

This is what it looks like:

```
ZTEST                                LS026                                23-11-03
                                      A0                                    10:14:44

Select one of the following:
```

- 1. Create test data generator program ZTSTGEN
- Function for file compare views
- 2. Create file compare views for a pair of files ZCMPGEN
- Functions for scripted tests
- 3. Settings for scripted test CALL ZSCRADMIN
- 4. Work with scripts - conversion, run, statistics CALL ZSCRIPT

- 90. Signoff SIGNOFF

Selection or command

===> _____

F3=Exit F4=Prompt F9=Retrieve F12=Cancel
F13=Information Assistant F16=System main menu

The test data generator and the file compare functions are set up as commands while the administrator functions and the functions for scripted testing are set up as program calls from the menu.



ZTEST User documentation

Configuration options

Functions for scripted testing

Use option 4 on the menu for this. "Settings for scripted tests"

This is what you will see:

```
LS026      ZTEST Script function settings      20-06-08
AO         V1.000                               15:35:01
-----
PC folder for macros:  \Z_Scripts
IFS folder for macros: /home/ao
Database Journalled Y/N: Y  Name: CUSJRN  Library: AO
Script file:          Name: QTNSCRIPT  Library: *LIBL
-----
F3=Exit no updates  Enter=Continue
```

- PC folder for macros
This is your way to tell the system about where you want the ACS macro recorder to store your macros. The folder is created if it does not exist. You will have to tell ACS (when recording) to place the macros here.
- IFS folder for macros
This tells the system where to look for macros when converting them to TNAPI scripts. The folder is created if it does not exist.
- Database journalled
A simple yes-no question. Only if the database tables are journalled to the journal you specify (name and library) is it possible for the function to view database changes for a script to work.
- Script file
This is where the converted macros as well as any scripts you may enter manually are stored. This is a simple source file. If it does not exist it is created.



ZTEST User documentation

ZTSTGEN – Test data generation

What is this used for?

Having enough test data is difficult. Having good enough test data is even more difficult.

One solution might be to copy and possibly anonymize production data. But there is always a risk that there is not enough variation in the production data to cover all cases – especially when the changes performed includes introducing new values or new columns.

Another problem is when there is a new application (or part of) that means that new tables are created. In this case there is no production data to copy.

What it does:

You pick a file and for each column in the file you describe how you want the data for that column to be generated.

You also specify how many records you want to be created for the file when the generation is run.

The function then – based on your input – generates and compiles the source for an RPG program that when run creates the number of records you want and is based on the rules you have set.

As for the rules and the options you have – we will come to that later.

What is required:

You will need:

- The ZTEST library in your library list
- An RPG compiler
- SQL Preprocessor
- And access to SEU (for certain options)



ZTEST User documentation

How to use the function

From the ZTEST menu or from a command line call and prompt the ZTSTGEN command.

Generate testdata (ZTSTGEN)

Type choices, press Enter.

File name	_____	Name
Sourcefile	<u>QRPGLESRC</u>	Name
Source member	_____	Name
Source library	_____	Name
Retrieve previous definition . .	<u>N</u>	Y, N
Store definitions	<u>N</u>	Y, N

- File name: A file in your library list. This is the target.
- Source file: This is where the generated RPG code is stored.
- Source member: The name you want for your source member. Also name of
compiled program.
- Source library: The library for the generated source.
- Retrieve previous: If this is set to "Y" the function will search for a previous definition that has been used for the file and fields used here. If that is found the values used before will be placed on the detail screens.
- Store definitions: Means that the selections you enter are stored in a file and can be retrieved when you generate next time.
- Nothing strange here.



ZTEST User documentation

A simple example:

```

Generate testdata (ZTSTGEN)

Type choices, press Enter.

File name . . . . . DEMOPFZ      Name
Sourcefile . . . . . qdemo       Name
Source member . . . . . DEMO01   Name
Source library . . . . . ZTEST    Name

```

I have a file named DEMOPFZ. For this I want to create a program DEMO01 and have the source created in source file QDEMO in library ZTEST.

The entered data is validated so that the file exists, source file exists and so on.

Default is for the source member to be overwritten if it existed before.

Next you will be presented with the following screen that lists the fields in your file:

```

Select Fields for which to apply generation rules      20-05-13
                                                    11:51:00

Number of records to generate: _____ Record Format: RDEM

Sel  Field Name  Kfld  Field Text
--  -
--  FIELDSTS     STATUS FIELD
--  FIELDDDRC    TO BE DERIVED
--  FIELDTXT     GENERIC TEXT
--  FIELDQTY     QUANTITY FIELD
--  FIELDPRC     PRICE FIELD
--  FIELDAMT     AMOUNT FIELD
--  FIELDDDTE    DATE FIELD
--  FIELDTME     TIME FIELD
--  FIELDSTP     TIME STAMP FIELD

-----
Press F10 to create generator program
Press F3  to quit

```



ZTEST User documentation

You will need to:

1. Enter how many records that the generator program shall create when it is run.
2. Indicate (with an "X" or whatever nonblank character) the fields where you want to specify criteria for how the data shall be generated.

After you have entered the criteria for all fields you will return to this screen. If you are satisfied with what you have entered use the F10 key to generate and compile the source code.

The generated program will not run at this point. You will have to call the program yourself.

Let us assume that we entered an "X" for each and every field above.

Also say we wanted a thousand records created when the program runs.



ZTEST User documentation

Details per field

This screen is displayed for each field that has been selected from the list.

```
                Select fields and conditions                20-05-13
                                                         11:56:55

Field  FIELDSTS
Text:  STATUS FIELD
Length: 00001 chars          digits    decimals
Generate as:
 Defaults based on field type
 Random character value A-Z
 Numeric value in the range of
                0          0
 Retrieved field _____ from file _____
 Select from the following alfabetic values:
                _____
                _____
 Current number from _____ step _____
 Date in the range _____ - _____
 Time in the range _____ - _____
Eval:
Source in: *LIBL/QFLDSRC      mbr _____ Edit:  _
Pattern:  _____

F12=Back
```

To the left is the “Generate as” selector. For each of the options there may be one or more choices.

Default

This means that the values for the field always will be the same. Typically, blank for character fields and zeros for numeric fields.

Random character

Each position in the character field will be filled with a character A through Z which is generated in random. You can use this to verify that the size of the field (if used to the max) fits on printouts and screens.

Numeric value

The field will get a value in the range from the from-value up to the to-value.



ZTEST User documentation

Retrieved field

The program will draw the RRN from the field you specify and read that record. It will then move the specified field value to your target field. Be aware that the field name should be unique within the generated program. It will still work if the names are the same but only if the data types and lengths are also the same.

The file name is checked so that it exists in your library list.

The field name is also checked so that it exists in the file you selected.

NOTE: While you can generate data for a target file in the QTEMP library you cannot use a QTEMP file as the source for retrieved fields.

Select from value

The program draws a random number from 1 to 5 (or the number of values you have specified) and moves that constant value to the target field.

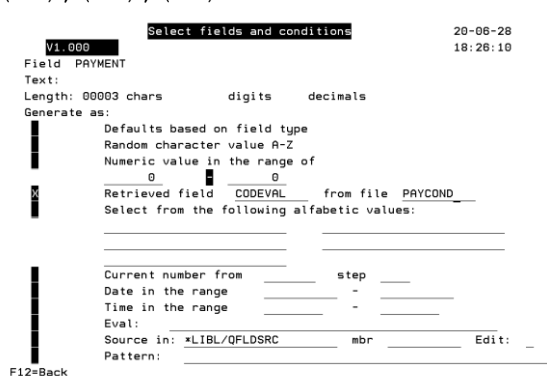
So, what if you have more than five values to choose from for a field?

In this case the recommended solution is as follows:

- Create a table using SQL
You can use interactive SQL or the “run SQL script” function in ACS or any other way. Create it with one field with the same type (and preferably length) as your target field.
- Enter data interactively in SQL for as many values as you need.
- Use the **retrieved field** option to populate the field from randomly selected values from the table. Now you can have as many values as you like.

Example payment conditions field.

```
CREATE TABLE MYLIB/PAYCOND (CODEVAL CHAR (3 ) NOT NULL WITH DEFAULT)
INSERT INTO PAYCOND (CODEVAL) VALUES (20) , (30) , (40) , (50) , (60) ,
(70) , (80) , (90)
```





ZTEST User documentation

Current number

A sequence number is drawn and moved to the field. You can select any starting number and increment that you like.

Date in range

The program calculates the number of days between the “from” date and the “to” date. It then draws a random number that falls in that number of days, increments the from date with that number and places the resulting date in your field. Enter the dates in YYYY-MM-DD format.

Alternatively, if you have a field that is suitable as a start or end value – enter the field name preceded by the ampersand “&” sign.

Note that the target for a date calculation can be any type of field! It can be a proper date field but also a numeric or character field of sufficient length i.e., 8 digits for numeric fields and 10 characters for character fields. This rule also applies to time fields (see below).

Time in range

This works in the same way as for dates. It only uses time values instead. Enter the times in HH:MM:SS format.

Eval

Here you can enter a small snippet of own code. Assume that you want the field “Value” to be calculated as $\text{Quantity} * \text{UnitPrice} * (1 - (\text{DiscountPct}/100))$ you enter this as :

```
&QUANTITY * &UNITPRICE * (1 - (&DISCOUNTPCT/100))
```

This is the easiest way that can place your own code in the generated program. The {fieldname =} part is assumed so all you need is code that defines or calculates a value.

For all field names in the formula that exist in the target file (the one that the data will be written to) place a “&” sign before the field name. This will cause the system to precede the field name with the qualifier for the output data structure.



ZTEST User documentation

Source in

Here you specify a source member that contains a snippet of code that you want to use as an /COPY member to calculate this field. You specify the source file and the member name and if you enter an “X” or a “Y” in the EDIT field you can also create the source or edit existing. Note that definition specifications are not allowed here – only fixed form or free form procedure specifications.

One example of such source is this case:

I am creating a program that will generate X number of order lines for order headers that already exist. The first line number within an order has to be 10 and then incremented with 10 for each new order line for the order number. How do I do that? Source in is the answer:

X	Source in: MYLIB/FLDSRC	mbr	FLINE	Edit: _
---	-------------------------	-----	-------	---------

The source that I entered myself is like this:

```
DsOut.ordline=*zero;
exec sql select max(ordline) into :DsOut.ordline from ordline where
orderno = :DsOut.orderno;
DsOut.ordline+=10;
```

Note that all field names that exist in the target file need to be qualified. This is because they are defined in a qualified data structure “DsOut”. If you do not qualify the field names, the compile will fail.

Meaning that I first set the result field to zero.

The I use SQL to find the highest line number for the order number in question. If no line exists, the line number will remain as zero.

Finally, I just increment whatever the line number has by 10 and so creates a new unique line number.

In the generated program is implemented as a simple “/COPY” statement.

Pattern

For character fields only. Enter “!” for alphabetic positions and “&” for numeric positions.

Any other characters where you want a constant. For example, the Swedish car registration numbers are in the format XXX999X – three alphabetic followed by three numeric and finally (recently added) an additional alphabetic. This would be entered as “!!!&&&!”.

In Finland they have a similar system but with a hyphen in between and no additional alphabetic. This would be “!!!-&&&”.

When all criteria have been entered

You will see the screen with the field list again. If you want to change anything select that field with an “X” and do the changes.



ZTEST User documentation

Creating the program

Press F10 and the program source is generated and compiled.

If there are errors?

You will get a message if the compile fails. If anything goes wrong, it most likely has to do with your own code snippets. Check the compiler listing and you will be able to sort it out.



ZTEST User documentation

What the code looks like

Declarations

```

- //*****
  // Program generated by ZTEST test data generator
  // Output file: ORDLINE
  // Generate:      500
  // Based on entered criteria (see below)
  //*****

  ctl-opt dftactgrp(*no) bnddir('ZTDGEN');
  // Files used to fetch values
  dcl-f ORDHEAD disk rename(ORDHEAD:ORDHEA_);
  dcl-f PROD disk rename(PROD:PRO_);

  // Structure for output data
  dcl-ds dsout extname('ORDLINE') qualified end-ds;

  // Copy prototypes for procedures for field generation
/COPY CRANDOM
  // Find highest record number for related files
  dcl-s RORDHEAD zoned(7:0);
  dcl-s RPROD zoned(7:0);
```

Comments

A comment block is included that shows file name, number of generated records and some more.

Control options:

We will be using a bunch of external procedures. Hence the binding directory. If you – in included code wishes to use external procedures in modules or service programs include these in the binding directory. Also, in this case edit the member CRANDOM (included with the product) and add the procedure prototypes for included procedures.

File-spec:

Any file where we fetch fields gets a file spec. If the record name is the same as the file name (typically for files created by SQL) the record format is renamed.

DS-spec:

The output file is declared as an external data structure. Note that the structure is now qualified which means that all fields need to be addressed with “DsOut.” before the field name.

/COPY

All prototypes and internal fields for the random generation procedures are in the copy source.



ZTEST User documentation

Record number field

For each linked file where data will be retrieved a variable is defined where the total number of records in the file are retrieved for later use.

Initial processing

```
ZSeed(); // Initiate random number generator
clear dsout; // Initialize output data structure
exec sql set transaction isolation level nc; // Do not use commit

RORDHEAD = Zrecnum('ORDHEAD'); //Find highest RRN
RPROD = Zrecnum('PROD'); //Find highest RRN
```

The starting point (“seed”) for the random generator is set.

All fields in the output format are initiated to defaults.

Journaling is bypassed for the output file.

Number of available records in (each) linked file is calculated.

Main processing

The program runs in a loop the number of counts you have specified.

For each field – other than the fields where default values are selected a small piece of code including call to a random number routine is executed.

As an example, if we want the quantity to field to have a value between 7 and 18 the code looks like this:

```
dsout.QTYORD=(Zrand * 18) +7;
```

For the calculated field (amount) the field is assigned whatever value we specified in our code snippet:

```
dsout.QTYREM=
DsOut.QTYORD - DsOut.QTYDEL;
```

Any included source for a field would have to include the target field name but in this simple case where the code snippet only resides on our screen input the program grabs the field name automatically.

Finally

The data is inserted using an SQL insert.



ZTEST User documentation

Resulting data

FIELDSTS	FIELDRC	FIELDTXT	FIELDQTY	FIELDPRC	FIELDAMT	FIELDOTE	FIELDTME	FIELDSTP
4	X3	AEUR#HECWGJGSSDBBTGWN@OTII#UV8K#A8BY#DOBLLCGMJFYS	24	109.84	2636.16	2020-05-02	08.41.32	0001-01-01 00:00:00.000000
9	VEYRON	#QEATEHPO@M#0EJWZZ@PN#LVBICL@ASW#U#E@WUWIMDSJIVA	24	572.16	13731.84	2020-05-09	11.15.54	0001-01-01 00:00:00.000000
9	AVENSIS	Y8D#YIP#KH#NQH2UEVLGX#JXWZTWDWIDIIGRAWOSUZSTIJII	15	298.78	4481.70	2020-05-03	16.03.40	0001-01-01 00:00:00.000000
9	TIGUAN	GBPJAQDVZY@#BB@YVCT@MEYJFOAWU#DVSBAJFUUVKROEVC#	9	293.93	2645.37	2020-05-02	07.16.31	0001-01-01 00:00:00.000000
9	TIGUAN	KO#KP#IKERQFJWMAEYDAR#VQOQPCJB#NVJ#YKJXV@PW@F@WZ	21	365.46	7674.66	2020-05-08	15.58.10	0001-01-01 00:00:00.000000
4	VITESSE	GLSUDFJIHOAPFEMHEN#DDYQDGDULVROVYSXMMARXXGGJESCYN	21	100.61	2112.81	2020-05-02	15.15.16	0001-01-01 00:00:00.000000
1	TIGUAN	ORLB@TXMVCIFFLHFMCBCLWQ#ZEYOMAHJTYUJCDYFSJUKW	3	134.98	404.94	2020-05-05	11.32.23	0001-01-01 00:00:00.000000
4	CLIO	VZDBWBPUFVW#CHELIXUK#GURCYFX#B#WJCGAJCH#VULVHK	21	301.20	6325.20	2020-05-12	15.31.20	0001-01-01 00:00:00.000000
1	VEYRON	IAGCTWUFEQJRAYTDGHRVASMZMZMPKLB@DGCNNUIEUZ@#XG@YI	9	101.34	912.06	2020-05-04	10.13.51	0001-01-01 00:00:00.000000

- FIELDSTS is a 1, 4 or 9.
- FIELDRC has been fetched as the field CARMODEL in file CARS2 based on a randomly calculated record number.
- FIELDTXT has been created as a random string of characters.
- FIELDQTY is a random number in a range.
- FIELDPRC is also a random number in a range.
- FIELDAMT is calculated as FIELDQTY times FIELDPRC.
- FIELDDTE is a random date in May of 2020.
- FIELDTME is a random time in the range of 6 AM to 4.30 PM.
- FIELDSTP has not been specified and so is set to the default value.

What if I have special requirements for fields – such as external procedures?

This is also easy to accomplish but requires a few extra steps.

1. The first means that you have to tweak the binding directory that is used by the generator. This is named ZTDGEN and is located in the ZTEST library.
As installed it lists only one object – a module named ZRANDOM that has a lot of functions for the random generator and for the various field types and rules.
If you have a module with a procedure that you will need or a service program just add this to the binding directory. (WRKBNDDIRE is a useful command for this).
2. Next use the option for own code (the one that creates a “/COPY” in the generated program and add code to call the procedure that you need. For example, if the voucher numbers for financial transactions are created by calling a procedure “Voucher” with transaction type as parameter your code would look like this:

```
Vchnbr=Voucher (TrnCde);
```


Easy so far?
3. Then when all field rules are entered press F10 to generate the source code and compile it.
4. Then the compile will fail! Why?? The reason is that you will need to add the procedure prototype in the definitions section of the program. Like

```
Dcl-pr Voucher Zoned(5);  
    TrnCde Char (2);  
End-pr;
```



ZTEST User documentation

If you do manual changes to the generated program

This is totally fine but there may be at least one minor thing that you must remember:

The program is compiled as an embedded SQL program and for the /COPY statements that happen to use SQL to work there is one compile parameter that needs to be set:

- RPGPPOPT (RPG Preprocessor options) should be set to “*LVL2.”

When you compile using the F10 key the package program takes care of this but your default settings for the compile command may not be the same.

Other potential issues

The generator program does not verify that your copy source or your “Eval” source is correct. Any errors there will only show up at compile time.

Final tip

Make use of the option to save the selections by field when creating even modestly complicated generator programs. It is so easy to retrieve (and possibly alter) your previous selections, and this can save you a little bit of time.

ZCMPGEN – Compare versions of files.

Important note!!:

For this function to work properly the files involved need to have unique keys defined by some method. Be that SQL-indexes, “logical files” or (heaven forbid) in the DDS of the “physical file”. This is what links the files together.

What is this used for?

This function is designed to provide help for the case when you have to compare two output data sets and identify any differences.

It could be that you have two versions of the program that creates the data or that a range of programs create it. It may also be the case that you have changes in the data definitions.

What it does:

After you have entered various criteria, the function creates SQL code and the following database objects:

Merge file

The “merge file” contains all columns for the two files where the designated key fields match.



ZTEST User documentation

The file is created as a SQL view meaning that it does not use any significant amount of storage. You can regard this as a compiled SQL statement that joins data from the “old” and the “new” file in one record.

“Miss files”

These views (two of them) hold the records where there is no match for the key fields in the other file.

The first one has the data where the key values exist in the “old” file but are missing in the “new” file.

The second file has the data where the key values exist in the “new” file but are missing in the “old” file.

Difference file

This last one contains data where specified (non-key) fields have different values in the “old” versus the “new” files. If any field designated as a compare field differs the data is included.

Additionally, you can specify that the file has fields for the value of the difference. This can be specified as an absolute value or if desired as a percentage.

What the function does not do:

There is no reporting tool included with the tool. As far as the compare function is concerned it has done its job when the views are created.

You should be able to analyze the contents using commonly used query tools or with SQL queries.

There are many reasons why this is so. One is that once you as an example know that certain field values differ between file versions you have lots of paths to take when you need to take the analysis to the next step. The tool gives you easy access to the data and that should give you quite a lot of help.

What is the performance impact of using the tool?

The process of creating the SQL views is a very quick and easy one.

SQL views take (as stated before) very little space.

They also have no related access path that has to be maintained by the system so there is no performance penalty for keeping these views on board.

Obviously when you access the data this uses some computer power depending on what you are doing with it.

Accessing the views is handled by the systems database optimizer which does a pretty good job of accessing data in the most efficient way possible.



ZTEST User documentation

How to use the function

First

Make sure that your library list contains the library that has the objects for the function. By default, this is the ZTEST library.

Enter the command parameters as follows:

```

Compare versions of same file (ZCMPGEN)

Type choices, press Enter.

Old file . . . . . _____ Character value
Old file library . . . . . *LIBL _____ Character value
New file . . . . . _____ Character value
New file library . . . . . *LIBL _____ Character value

```

Main screen

You do all your selections and definitions through options on the main screen.

```

Create compare files                                20-05-28
                                                    16:39:47

Steps:

Select  Action                                     Status
-       1.Select output files and source           PENDING
-       2.Select unique access path for            PENDING
        retrieval of key fields (optional)
-       3.Select key fields for join              PENDING
-       4.Select fields to check for               PENDING
        differences
-       5.Generate source and compile             PENDING

F3 = Exit

```

There are five steps that need to be performed in sequence.

One is totally optional (second step where you select an access path) and one is only used when you create the view that contains the data where field values do not match (fourth step).



ZTEST User documentation

When one step has been completed the select input field is blocked and the status changed to completed. If you skip the access path step and select key fields manually the status for option 2 will be “bypassed”.

When all is marked complete just exit the function with the F3 key.

Select output files and source.

```
Create compare files                                20-05-28
                                                    16:46:16

Step 1: Define output data name and locations

Merge file:      Name          Library   SQL srcf   *CURLIB
Missing new :   *NONE         *DTALIB  QSQLSRC   *CURLIB
Missing old :   *NONE         *DTALIB  QSQLSRC   *CURLIB
Differences :   *NONE         *DTALIB  QSQLSRC   *CURLIB
```

You will have to define the name for the merge file. This is the view that combines all fields from the old file and the new file where the keys you select later match.

The view for records in the old file that are missing in the new file, the one for records in the new file that is missing in the old file and the view for data where keys match, but specific values differ between the files can be omitted if you like.

The values for file library, source file and source library are saved when the files are created so that your preferred defaults are retrieved next time you use the function.

For each view that you want created you need to provide:

- A valid name (10 characters).
This will be the name of the object as well as the source member that is generated.
- A library where the view shall be created. The default value here is *DTALIB which means that the view is created in the same library as that where the “old file” is located.
- A source file where you want the SQL code created. If the file does not exist it will be created when the SQL source is generated.
- The library where the source file is placed. If you specify *CURLIB this will be your own current library.

Do note that you will not have to re-create the files just because you have run a new test with new data. The views created will always show the current situation!



ZTEST User documentation

Example:

I have to “order files” named SAMPLEOLD and SAMPLENEW. If I want to create views for these this is how I might enter the data:

```
Create compare files                                20-05-28
                                                    16:46:16

Step 1: Define output data name and locations

Merge file:      Name      Library  SQL srcf  ZTEST
                 MERGE0528  *DTALIB ZSQLSRC
Missing new :    MSNEW0528  *DTALIB ZSQLSRC  ZTEST
Missing old :    MSOLD0528  *DTALIB ZSQLSRC  ZTEST
Differences :    MDIFF0528  *DTALIB ZSQLSRC  ZTEST
```

I want the views in the same library as the data. I have a source file in library ZTEST named ZSQLSRC and this is where I want the source to be generated.



ZTEST User documentation

Select unique access path.

When you run this option, the purpose is to try and automate the selection of key fields.

The function looks for an access path (index) with attribute “unique” and displays all of these.

Choose one and the key fields will be pre-selected in the next step.

You may skip this step and select the keys manually.

In my example there is one such key file as shown below.

```
      Create compare files                                20-05-28
                                                         17:00:49

      Step 2: Select primary key file
                Option 1=Select key fields from this file
Old          Enter without select=Choose key fields manually
? File      Text
_ SAMPLEOLDL Sample old LF
```

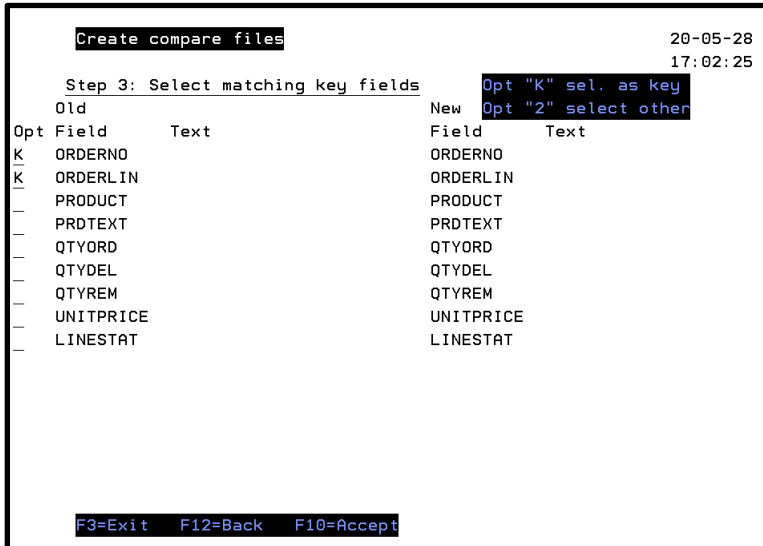
Select by enter a “1” in the option field (under the “?”) and press enter.



ZTEST User documentation

Select matching key fields.

That is define what fields in the two files that shall be used to join the records.



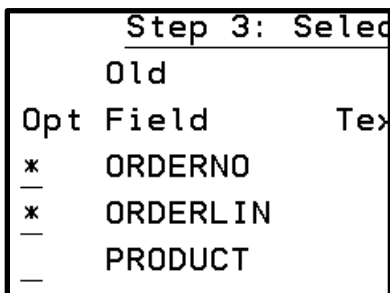
You have the following available choices:

- If you select an access path in the previous step the key fields from that access path are already pre-entered with a “K” in the options field.
- You can select other keys by removing the “K” and enter a “K” on the line for a key field that you prefer. Do note however that for the matching to work you need the actual “proper” keys for the files – that what uniquely identifies a record.
- By default, the key field from the “old” file is set to match a field with the same name in the “new” file. If you want the field to match another field just enter a “2” in the options field and press enter. See “selecting other field to match” shortly!

If you have messed things up and want to start again just press F12 and start over!

When you have defined your key fields...

Press enters and the lines with “K” will be seen with an asterisk:



Press F10 when you are happy with your choices.



ZTEST User documentation

Select fields to check for differences.

This again shows a list of all fields in the files.

```
Create compare files                               20-05-28
                                                    18:27:50

Step 4: Select function fields
Option A=Absolute value P=Percent C=General compare 2=Sel.field
Old
Opt Field      Text                Type      New/Compare
  Field      Text                Type      Field      Text
- LINESTAT
- ORDERLIN
- ORDERNO
- PRDTEXT
- PRODUCT
- QTYDEL
- QTYORD
- QTYREM
- UNITPRICE

F3=Exit  F12=Back  F10=Accept
```

The purpose here is to enter conditions that will be used for the view that shows record data where non-key fields have different values in the two files.

All records in the *merge* file where the fields selected differ will be included.

In addition, those fields where you select "A" for absolute value or "P" for percent will cause a calculated field to be included in the view.

Assume that I want to check that the same product is used, that line status is the same and also for all the quantity fields I want to see the difference (if any) plus a percent difference for the unit price.

This is what I would enter:

```
Old
Opt Field
C  LINESTAT
  ORDERLIN
  ORDERNO
  PRDTEXT
C  PRODUCT
A  QTYDEL
A  QTYORD
A  QTYREM
P  UNITPRICE
```



ZTEST User documentation

When I press enter the screen will look like this:

```
Create compare files                                20-05-28
                                                    18:33:48

Step 4: Select function fields
Option A=Absolute value P=Percent C=General compare 2=Sel.field
Old
Opt Field      Text                               Type      New/Compare
--
-- LINESTAT
-- ORDERLIN
-- ORDERNO
-- PRDTEXT
-- PRODUCT
-- QTYDEL
-- QTYORD
-- QTYREM
-- UNITPRICE

--                               CHR
--                               ABS
--                               ABS
--                               ABS
--                               PCT

--                               LINESTAT
--                               ORDERLIN
--                               ORDERNO
--                               PRDTEXT
--                               PRODUCT
--                               QTYDEL
--                               QTYORD
--                               QTYREM
--                               UNITPRICE

F3=Exit  F12=Back  F10=Accept
```

The type column has been changed to reflect my selections.

If I want to restart and backout of any changes I have made I just press F12 and start again with this option.

Selecting another field from "new" file (option 2)

Assume that I want to compare product to product text instead of product.

I enter a "2" in the options field:

I will see a window with fields to choose from. First fields of same type and length, next fields of same type and finally all other fields.

```
Select matching field
Position to:  PRODUCT
Field type :  _
1=Select
? Field      T Len  Dg Dc  Text
--
-- LINESTAT   A 00001 00 00
-- ORDERLIN   A 00006 00 00
-- ORDERNO    A 00006 00 00
-- PRDTEXT    A 00020 00 00
-- QTYDEL     S 00007 07 07
-- QTYORD     S 00007 07 07
-- QTYREM     S 00007 07 07
-- UNITPRICE  S 00011 11 11
```

Enter a "1" for the matching field to choose and press enter.

OR F12 to exit without changing anything.

When all the compare fields have been selected press F10 to save and exit.



ZTEST User documentation

Generate source and compile.

```
Create compare files                                20-05-28
                                                    18:42:39

Final step Create output source - or backout

All selections are now complete.
Press F10 to generate and compile SQL source or
press F3 to exit without saving.

F3 = Exit - no generation F10 = Generate and compile views
```

At this stage there is nothing more to select.

You have two options:

- F3 will exit the function without saving anything.
- F10 will create the output SQL source and invoke the SQL function to create the views.

When you see the main screen with the text “COMPLETED” for the generate option all should be well.

```
File . . . . . ZSQLSRC
Library . . . . . ZTEST

Type options, press Enter.
2=Edit          3=Copy  4=De
8=Display description  9=Sa

Opt  Member      Type
---  ---
   MDI0528      SQL
   MNE0528      SQL
   MOL0528      SQL
   MRG0528      SQL
```



ZTEST User documentation

What has been created?

In this case (the file names here are my own!)

MRG0528 a view that combines ALL fields from both files where the keys match.

- The OLD fields have the name prefixed with "OLD_"
- The NEW fields have the prefix "NEW_"

OLD_ORDERNO	OLD_ORDERLIN	OLD_PRODUCT	OLD_PROTEXT	OLD_QTYORD	OLD_QTYDEL	OLD_QTYREM	OLD_UNITPRICE	OLD_LINESTAT	NEW_ORDERNO	NEW_ORDERLIN	NEW_PRODUCT	NEW_PROTEXT	NEW_QTYORD	NEW_QTYDEL	NEW_QTYREM	NEW_UNITPRICE	N
A0001	000001	SK116088R-NEM	Kantappel	1.25	0.25	1.00	0.75 A		A0001	000001	SK116088R-NEM	Kantappel	1.15	0.25	0.90	0.95 A	

The merge file is always created.

MNE0528 records in OLD missing in NEW

A simple view that shows all records that do not have a match in the other file.

MOL0528 records in NEW missing in OLD

A simple view that shows all records that do not have a match in the other file.

MDI0528 records with differences in selected new key fields

This has one record for every case where records where the keys match but one or more of the selected non key fields have different values in the OLD file compared to the NEW file.

In addition – at the end – you will have the additional calculated fields that show differences for numeric fields in absolute value or as a percent value.

These function fields have the prefix "DIFF_" for absolute value and "PCT_" for percent:

DIFF_QTYDEL	DIFF_QTYORD	DIFF_QTYREM	PCT_UNITPRICE
0.00	0.10	0.10	2.2857142857142857100

In this case quantity ordered and remaining have differences. Also, the unit price has a 2.3% difference.

If the underlying data changes the data derived from the view is also changed.



ZTEST User documentation

Scripted testing

What this does

This function gives you the possibility to:

- Record any actions that you do on a 5250 screen.
- Submit what you recorded to run in batch.
- Analyze what happened when the script ran and verify the results.
 - Visual i.e., what screens were displayed.
 - Database i.e., what updates were performed.

How this works

The function uses the TNAPI framework that is developed and provided by Albert York. This is the part that makes 5250 functions run in batch.

On top of this is the step from the recorded macro to an executable script.

There is also the analysis layer that has been added to what the TNAPI provides.



ZTEST User documentation

Required setup.

You will need to do a few simple steps before getting started.

The first is to create a folder on your desktop. This is the location where the recorded macros will be stored after you finish recording.

The second task is to create a folder on the IFS. This is used in the conversion process as this can read from the IFS but not from data on your desktop AND the macros cannot (currently) be recorded to the IFS directly.

You will also need a source file QTNSCRIPT where the final scripts will be stored.

Obviously install of both the ZTEST and TNAPI libraries is required. See the package install document on how to do that.

Use the admin function for scripts and enter the values you want:

LS026	ZTEST Script function settings	20-04-16
AD		11:42:32

PC folder for macros:	_____	
IFS folder for macros:	_____	
Database Journalled Y/N:	Name: _____	Library: _____
Script file:	Name: QTNSCRIPT	Library: *LIBL

F3=Exit no updates Enter=Continue		

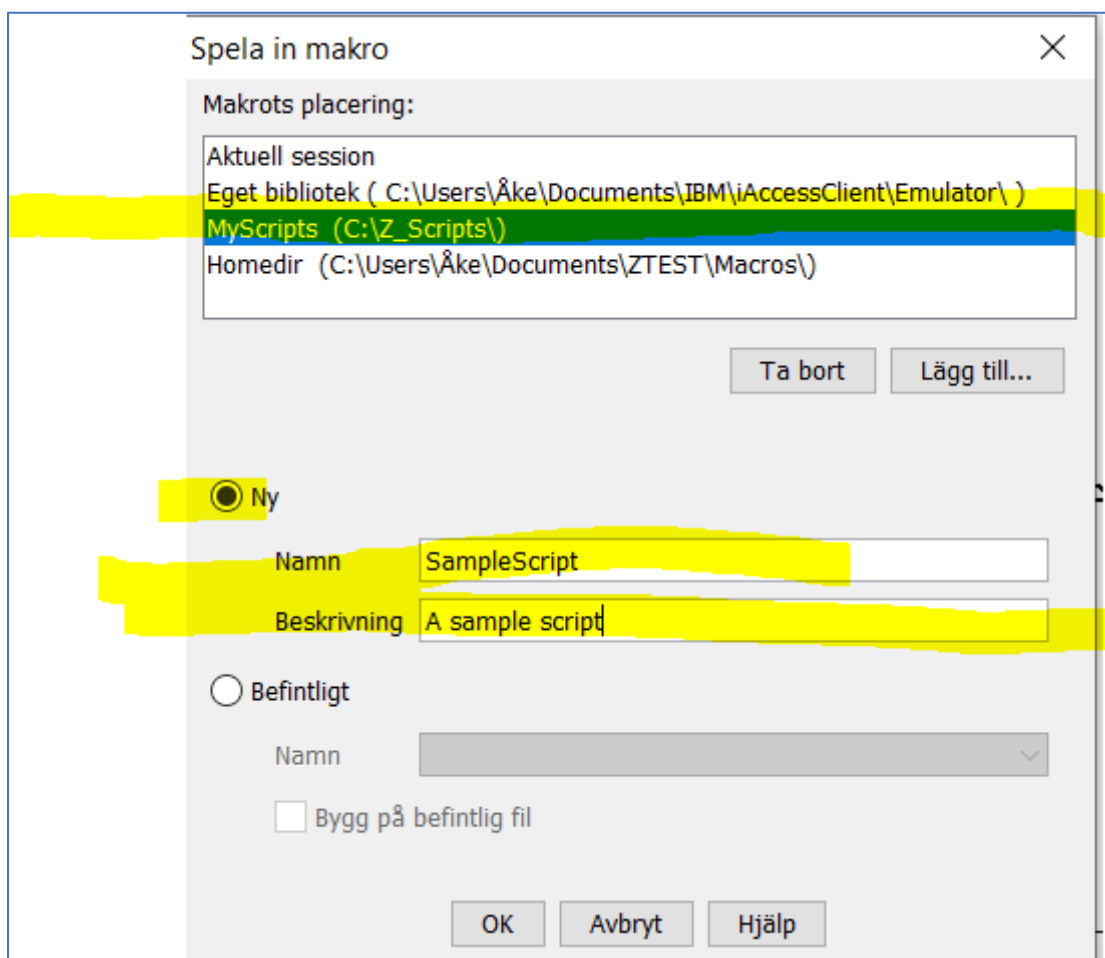
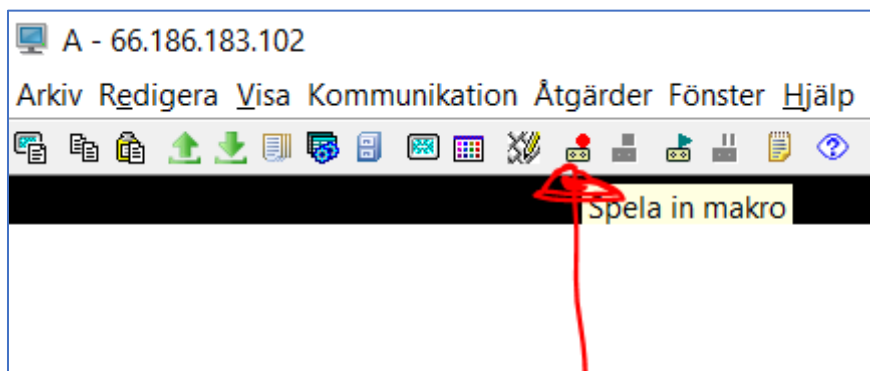


ZTEST User documentation

Recording a macro

Log on the IBM I and once logged in start recording a script.

Click the Icon to start recording:

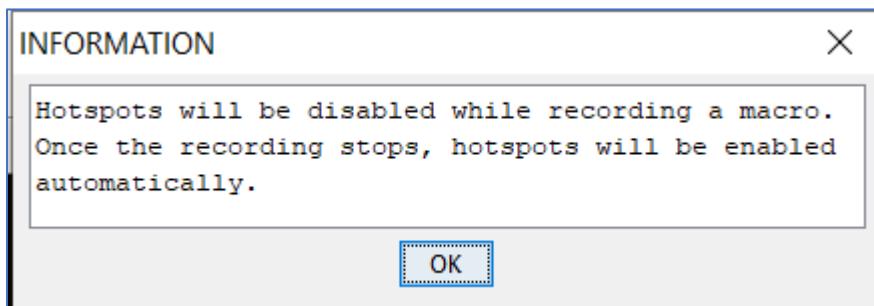


Then select the PC folder where you decided to store the macros. Give the macro a name and a description.



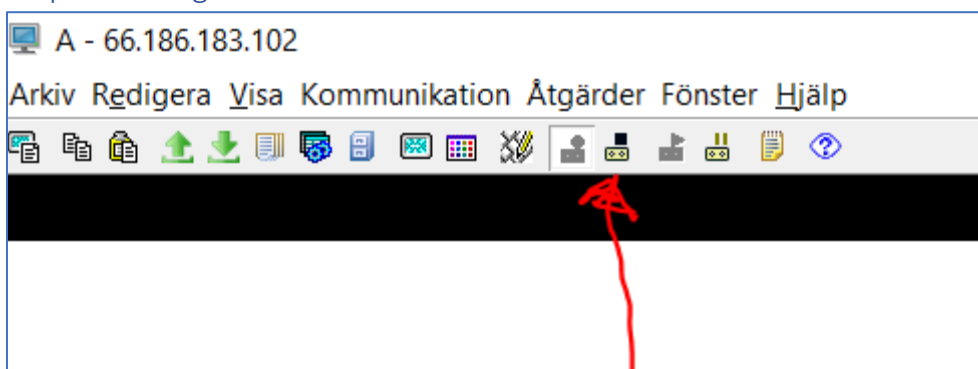
ZTEST User documentation

A window like the one below may appear.



Just click "OK" to continue.

Stop recording macro.



Click the icon to end recording.

Transfer macro(-s) to IFS

Use the ACS function to access the IFS or use any other preferred method to copy the recorded macro to the designated folder on the IFS.

Convert recorded macro to script file.

Run the menu option for macro conversion.

Why is this step necessary?

The format for the scrips that run through the RUNTNSCR function is quite different from the macro format that is used by ACS which uses the host on demand format and has a lot of extra information that is not used by the RUNTNSCR function.

For more details on the scrips see the documentation that comes with TNAPI!

What happens?

The program will show you all the macro files that are placed in the IFS folder you entered during the setup.

Any files that do have a corresponding script is marked to indicate this.



ZTEST User documentation

Check all macros that you want to have converted to scripts and hit enter.

If the script existed before it will be replaced.

A macro (beginning of it) looks something like this:

```
<HAScript name="customer_mnt" description="Add a customer" timeout="60000" pausetime="300" promptall="true" blockinput="tru
  <screen name="Skärm1" entryscreen="true" exitscreen="false" transient="false">
    <description >
      <oia status="NOTINHIBITED" optional="false" invertmatch="false" />
    </description>
    <actions>
      <input value="call customer[enter]" row="0" col="0" movecursor="true" xlatehostkeys="true" encrypted="false" />
    </actions>
    <nextscreens timeout="0" >
      <nextscreen name="Skärm2" />
    </nextscreens>
  </screen>
```

The " <HAScript" section is used to fetch the description which is also used as the member.

The rest of the descriptions for each display is converted as keystrokes and function keys for the script.



ZTEST User documentation

Working with scripts

This is available from the ZTEST menu or by calling the program ZSCRIPT.

When you run ZSCRIPT the screen looks something like this:

```
LS026      Work with scripts      20-06-11
AD         V1.000                  09:33:49

-----
Options:   2=Edit, 4=Delete, 5=View, S=Submit
           D=Work with data updates (last run)
           V=Work with screen dumps (last run)

Submit user: _____
Password:   _____

Opt Script Text
-----
  - CUSTOMER Customer maint
                Last run: 2020-04-28 11:57:32
  - CUSTOMER_M Add a customer
                Last run:
  - SAMPLE1    Alfa
                Last run: 2020-04-22 11:56:09
  - SAMPLE2    Beta
                Last run: 2020-04-21 11:12:00
  - SAMPLE3    Gamma
                Last run: 2020-04-21 11:12:01
  - TEST2      Customer click o tab
                Last run:

-----
F3=Exit    F6=Convert recorded ACS macros
```

The version number for the package is shown at the top.

At the top there are fields for submit user and password. This is used when you submit scripts for execution. This will be the user profile that the script is run under.

Each script (which exists as a source member in the script file you have defined) is listed together with information on when it was run last. When you check results (screen shots and database changes) this is the instance that you will be looking at.



ZTEST User documentation

Available options for the scripts

2 – Edit a script

This starts editing with SEU for the source member containing the script.

3 – Copy a script

You are prompted for a new name. The script is copied to a source member with this name.

4 – Delete a script

This removes the source member and the screen history for the script.

5 – View a script

This – again – starts seu but in browse mode.

S – Submit a script

Submitting a script calls a function that runs the script in a batch job.

This job has the same name as the script. The same job name is also found on the spool file (script log file) that is created.

One thing to note is that the actual job that runs the interactive function has another name since this is a special interactive job that is spawned by the RUNTNSCR function. If you look at the history log you will find this. This has to be so due to the mechanics of running 5250 screen processes in a batch environment.

D – View database changes

For this to work you must have the database tables affected by the function journaled to the journal that you entered on the setup screen.

There is also another “catch” for viewing database changes for scripts that have been submitted.

Although the running of a script is submitted to a job with the same name as the script this is not the job where any program invoked by the script processor is running. This has to do with the way the script processor communicates with the IBM I. When a socket communication is initiated, this starts a secondary job that mainly looks like an interactive job i.e., with the job name being that of a virtual device “QPADEVnnnn”. We cannot control the name of that job. It is what it is.



ZTEST User documentation

Because of this the journal entries retrieved are those based on:

- User (same as the user that ran the script)
- Time interval – entries from when the script job started until it ended.

IF the same user has other processes running at the exact same time and that performs (journalized) database updates you might see more journal entries than you expected.

You will see a screen like the one below with which journal entries have been recorded.

Type equals “UB” for record before update, “UP” for record after update, “PT” for record inserted and “DL” for deleted record.

To see details, enter “5” in the options field and press enter.

```
LS026                                     20-04-27
AD                                     View database updates for script 10:39:25
-----
Script:  SAMPLE1

?  File      Lib      Seq      Type  RRn
-  -
-  0000000000 0000000000 00
-  0000000000 0000000000 00
-  0000000000 0000000000 00
-  0000000000 0000000000 00
-  0000000000 0000000000 00
-  0000000000 0000000000 00
-  0000000000 0000000000 00
-  0000000000 0000000000 00
-  0000000000 0000000000 00
-  0000000000 0000000000 00
-  0000000000 0000000000 00
-  0000000000 0000000000 00
-  0000000000 0000000000 00
-  0000000000 0000000000 00
-  0000000000 0000000000 00
-  0000000000 0000000000 00
-  0000000000 0000000000 00
-  0000000000 0000000000 00
```



ZTEST User documentation

```
.....1.....2.....3.....4.....5.....6.....7.....8.....9.....10.....11.....12.....
Customer  NAME                STREET                STREET2                CITY
Number
123,456  XCUS                        B GATAN                TIDSTAMP                UPPSALA
*****  End of report  *****

F12=Cancel  F19=Left  F20=Right  F21=Split  F22=Width 80  Bottom
```

You will see all columns in the related record image as one line.

Press F20 to pan to the right and F19 to pan to the left.

Exit by using the F3 key.

V – View screen data

When the script is run all screen data is captured in a log file SCRIPTLOG with one member for each script name. The data from the most recent run is stored here.

Each “enter” moves you one screen forward. When you press F5 you see the script data that was sent to the screen.

Function key F6 - Converting recorded macros to scripts.

This function has been temporarily disabled in release V1.000. Will be restored when V1.001 is released shortly.

ACS macros have a totally different format than TNAPI scripts. This is why the macros have to undergo a conversion process before they can be used.

Step one for this is to load all your ACS macros into the IFS folder that you configured during the setup process (ZSCRADMIN – option 4 on the ZTEST menu).

- Start the IFS option in ACS.
- Navigate to the folder you defined/created.
- Transfer the macro files (suffix .mac) from the desktop system to the folder.
Use Ctrl + U to start this function.



ZTEST User documentation

When you have your macros safely in the IFS folder press F6

```
LS026      Convert ACS Macros to Scripts      20-06-11
AO         V1.000                             10:07:35
-----
Options:   2=Convert
Opt Macro
- /home/ao/customer_mnt.mac
- /home/ao/customer2.mac
- /home/ao/SampleScript.mac
- /home/ao/ZtestAdmin.mac
```

Enter a "2" before each macro that you want to have converted to a TNAPI script and press enter.

Exit the function by using the F3 key.



ZTEST User documentation

How TNAPI scripts are formatted

The scripts use two statements (there are some other options available, but these are the essential ones):

DATA Statement

This is where all values are placed as well as certain control sequences for cursor placement, field-exit, tabs and similar codes. The codes used here are those that do not cause data to be sent to the program.

SEND Statement

This sends the data that was entered. Here we need to provide a code that tells the program HOW the data is sent – that is if it should simulate the enter key, a function key or a page-down/page_up key.

Available cursor control codes for the “DATA” statement

Code	Meaning
%BS	Backspace
%DN	Down
%LF	Left
%RT	Right
%UP	Up
%DL	Delete
%DP	Dup
%ER	Erase input
%BT	Back tab
%FE	Field exit
%FP	Field plus
%FM	Field minus
%IN	Insert
%NL	New line
%TB	Tab

Example: If you have a form and three values should be entered the DATA statement might look something like this:

```
DATA First_value%TBSecond_value%TBThird_Value
```

Note that at this stage nothing is sent to the program!



ZTEST User documentation

Available control codes for the SEND statement

Code	Alternatively	Meaning
ENTER		Enter key
F1	F01	Function key 1
F2	F02	Function key 2
F3	F03	Function key 3
F4	F04	Function key 4
F5	F05	Function key 5
F6	F06	Function key 6
F7	F07	Function key 7
F8	F08	Function key 8
F9	F09	Function key 9
F10		Function key 10
F11		Function key 11
F12		Function key 12
F13		Function key 13
F14		Function key 14
F15		Function key 15
F16		Function key 16
F17		Function key 17
F18		Function key 18
F19		Function key 19
F20		Function key 20
F21		Function key 21
F22		Function key 22
F23		Function key 23
F24		Function key 24
PAGEDN		Page down (roll up) key
PAGEUP		Page up (roll down) key
ATTN		Attention key
SYSREQ		System request key
HOME		Home key
PRINT		Print key
RESET		Reset function (unlock keyboard)
REDRAW		?
HELP		Help key



ZTEST User documentation

Example:

Assume a case with data entry using DFU.

The goal is to use the UPDDTA command for a file and then change one value in the third record in the file.

This would contain the following steps:

1. Send the UPDDTA command.
2. Send three page_down keys (to scroll to the correct record)
3. Send one (or more) tab keys to move cursor to the correct field.
4. Enter the data.
5. Send the ENTER key.
6. Exit DFU by sending the F3 key.
7. Send a signoff command.

This is what the script would look like (in full):

The file used here is the “PROD” file in a development library.

1. DATA upddta ztest_dev/prod
Calls the UpdDta function – when sent
2. SEND ENTER
Sends the command above
3. SEND PAGEDN
Scroll down one record in DFU.
4. SEND PAGEDN
Scroll down one record in DFU.
5. SEND PAGEDN
Scroll down one record in DFU.
6. DATA %TBLOVELY BLOKE
Position to the second field in the format by sending a tab-key. Followed by the text “LOVELY BLOKE”
7. SEND ENTER
Send the above data.
8. SEND F3
Exit DFU by sending an F3 key.
9. SEND ENTER
Send one enter at end of DFU (this is where DFU shows number of changed/added records)
10. DATA SIGNOFF
Add SIGNOFF command to screen.



ZTEST User documentation

11. SEND ENTER

Finally send the signoff sequence.

Screen log when run:

```
Selection or command
==> upddta_ztest_dev/prod__
```

2020-06-28-19.35.20.456 SENT: <PAGEDN>

```

... Intermediate screens omitted as well as second and third "<PAGEDN>
WORK WITH DATA IN A FILE                               Mode . . . . . : CHANGE
Format . . . . . :  PROD_____                       File . . . . . :  PROD
*RECNBR:          3
PRODUCT:  ___103_
PRODTXT:  CHARMING_BLOKE_____
F3=Exit          F5=Refresh          F6=Select format
F9=Insert        F10=Entry           F11=Change

```

Note that the field "Product" was skipped over by sending the tab key – only PRODTXT has been affected. IF %FE had been used instead of %TB the product number "103" would have been blanked/zeroed out.

2020-06-28-19.35.21.246 SENT: <F3>

F3 key sent

```

                                     End Data Entry
Number of records processed
  Added . . . . . :          0
  Changed . . . . . :          1
  Deleted . . . . . :          0
Type choice, press Enter.
  End data entry . . . . . :    Y          Y=Yes,  N=No

```

Summary screen at DFU exit.



ZTEST User documentation

This shows you

- That the script language (at least in its basic form) is quite straight forward and easy to use.
- That you can actually both create and modify scripts manually by using whichever editor you prefer.
- But for complex sequences you get helped quite a lot by using the record and convert option.



ZTEST User documentation

Updating the package

Finding the updates

On the [download page](#) for ZTEST there is a description on the latest changes.

In case there are changes just download and install the product library again.