



Script Designer

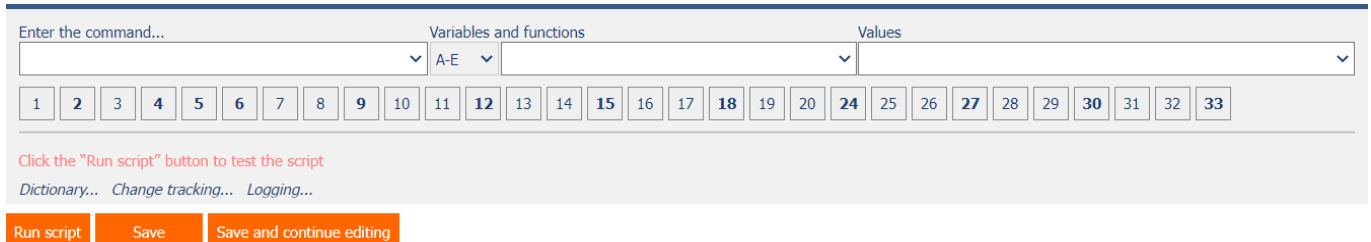
Framework NET Genium



Content

1	Script Designer	3
1.1	Enter the command... ..	3
1.2	Variables.....	4
1.3	Function.....	4
1.4	Values.....	4
1.5	Dictionary	4
1.6	Change tracking	4
1.7	Run script	5

1 Script Designer



The screenshot shows the Script Designer interface. At the top, there is a text input field labeled "Enter the command...". To its right are two dropdown menus: "Variables and functions" (set to "A-E") and "Values". Below these is a row of 33 numbered buttons, with buttons 1 through 23 being grey and buttons 24 through 33 being orange. Below the buttons is a red text prompt: "Click the 'Run script' button to test the script". Underneath are three links: "Dictionary...", "Change tracking...", and "Logging...". At the bottom are three orange buttons: "Run script", "Save", and "Save and continue editing".

1.1 Enter the command...

- A drop-down list with a selection of the type of command that will be inserted into the script at the selected location.
- A detailed description of the commands is given in the separate "Administrator's Guide".

1. Declare a variable
2. Add a value from the query
3. Save the variable back to the database
4. Add multiple values from the query
5. Cycle
6. Add a value
7. If (expression) assign a value
8. If (composite expression) assign a value
9. Go to the line
10. If (expression) go to the line
11. If (compound expression) go to the line
12. Raise Exception (Interrupt)
13. If (expression) raise an exception
14. If (compound expression) raise an exception
15. Send email
16. If (expression) send an e-mail
17. If (compound expression) send e-mail
18. Delete entry
19. If (expression) delete the record
20. If (compound expression) delete the record
24. New record
25. If (expression) a new record
26. If (compound expression) a new record
27. Save the value to the database
28. If (expression) store the value...
29. If (compound expression) store the value...
30. Store multiple values in a database
31. If (expression) save more...
32. If (compound expression) save more...
33. Comment

1.2 Variables

- Drop-down list with a selection of variables that will be available in the script for storing values. These variables can be used on individual lines of the script.
- A detailed description of the variables is given in the separate “Variables” manual.

1.3 Functions

- Drop-down list with a selection of server functions that can be called on individual lines of the script.
- A detailed description of the server functions is given in the separate “Server Functions” manual.

1.4 Values

- A drop-down list with a selection of database control identifiers that can be used on individual lines of the script.

1.5 Dictionary

- The dictionary displays a list of comments and interrupts used across all scripts throughout the application that relate to the following types of commands:
 - The “message” parameter of the “SCHEDULEBACK” server function in the script comment
 - The “message” parameter of the “TEFUIG” server function in the script comment
 - The “message” parameter of the “TESUIG” server function in the script comment
 - The “message” parameter of the “WRITEMESSAGE” server function in the script comment
 - Script abort
- Dictionary search is used to design the correct form of user messages and to maintain the consistency of nomenclature throughout the application.

1.6 Change tracking

- Change tracking displays a detailed report with all script changes made by individual users.

1.7 Run script

- The “Run script” button is used to run the script in test mode.
- Running a script simulates the progress of script processing, but does not store any values in the database or delete any records from the database.
- External functions called from the script in test mode are not executed – the value “true” in the “test” parameter is passed to the external function:

```
using System;

namespace NETGenium
{
    public class ExternalFunctions
    {
        public static string ngef(string id, string[] args, bool test, DbCommand cmd, DbConnection conn)
        {
            if (test) return "";

            switch (id)
            {
                // case "MyFirstFunction": return MyFirstFunction();
                default: return conn == null ? "" : conn.ExternalFunctionNotFound(id, cmd);
            }
        }
    }
}
```