NoiseMap fi✓e

Example Script Files

1. CONTENTS

2. ABOUT SCRIPT FILES

INTRODUCTION

Although the Script Interface gives access to the majority of NoiseMap functions, it isn’t a complete macro language. One issue is that although scripts permit users to define and use variables, there is no means of prompting for user input during the execution of the script other than yes/no/cancel options, so the user has to **set** values for any relevant script variables before the script is executed, or enter the values on the command line (which is the main approach in the following examples.)

Many scripts require a database name, scenario and combination to be set prior to running the script. A **Message Box** can alert the user where this is needed.

Scripts will probably fail if there is a typing error, such as in a scenario name. This can be annoying – for example when a script that you expected to take all night to execute fails after a short while because of such a typing error. It will be advisable to test out a script, e.g. by making a short trial run. There is no pre-run test function and in any case such a function would not able to pick up some types of error. You cannot perform any arithmetic operations on script variables.

The general-purpose scripts in this collection will help you to see what can be done with scripts and give you prototypes that you can use to develop more sophisticated ones. *For an explanation of the Script Commands, please see the separate NoiseMap Guide to Script Interface and Expressions.*

The source text of each of the scripts is provided in a file. We suggest you run some of them on the example model `manualexampledatabase.nmdb` to gain more insight into their operation.

Please note that this guidance is in draft and will contain some errors. Any feedback will be welcome.

SOME RULES ABOUT SCRIPT FILES

Script files must be written as plain text ASCII (ANSI) files. You must avoid the use of typographical symbols such as fancy quotes “” or apostrophes ‘’. Only use straight quotes like " and '. It is best to use a text editor such as NotePad or similar as these will not convert the standard keyboard symbols.

Although you can in theory type lines of any length, this is inconvenient when it comes to reading them. Where a command requires a lot of parameters, it is best to end the line at a convenient point with the & symbol, and then continue on the next line. This lets you enter a long command over as many lines as necessary.

Many commands have parameters that take things such as file names that may have spaces and other symbols in them. These ‘strings’ must be enclosed in quotes. Sometimes the parameters themselves may have quotes in them. To represent a quote within a quoted string, use a double quote, for example:

```
"This is a ""quoted"" string."
```

Or even:

```
"This string is ""quoted."""
```

SCRIPT FILE NAMES

Any valid Windows file name can be used. Script files can have the file type extension of .nms which NoiseMap will recognise.

SCRIPT VARIABLES

See the Script manual for information on script variables. You cannot perform arithmetic operations on script variables.

RUNNING A SCRIPT FILE

There are two main ways to run a script file

- □ From the `Run Script` menu option on the Graphical Screen File Menu
- □ From the Script Command Pane at the bottom of the Log/Results screen. This may be more convenient as the progress of the script will be shown in the Log/Results Screen

A script command can also be called from within another script command by using the `RUN` or `INCLUDE` command.

Note that at present NoiseMap does not look for script files in any particular location, and so you will probably have to type in the full path name (unless you set up a Windows Environment Variable containing this).

USING THE EXAMPLE SCRIPT FILES

We cannot be sure that these script files will work in every situation. Please use them advisedly.

SUPPLY INFORMATION FOR RUNNING THE SCRIPT

Scripts will require some information on what you want them to do. There are two ways of doing this.

- □ Provide the parameters in the command line, e.g.

```
run calcall "Base scenario"
```

- □ Set a user variable and then run the command, e.g.

```
set @scenario "Base scenario"

run calcall
```

Both systems have pros and cons. You will need to choose the more convenient when you are creating your script file. In the present examples, parameters are entered on the command line unless stated otherwise.

3. EXAMPLE SCRIPT FILES

CALCRTN "AREA" "SCENARIO"

This script calculates database contours *and* receivers for a named area and scenario. It assumes user is already connected to the required database. It calculates using CRTN 2005 method for all category combinations. It uses a surround of 2 tiles. Contours are 4m above ground at a 10 m grid spacing.

To use this script, connect to the required local flat-file or remote database and type (substitute the correct area and scenario names):

```
run calcrtn "Area" "Scenario"
```

Example of command line, showing use of quotes:

```
run "\\Cerver\Script files\calcrtn.nms" "Burton Estate" "With New Road"
```

LOADRESULTS "AREA" "SCENARIO" "COMBO"

This script loads *Receiver Results* from the currently-connected database. It assumes that the required results have already been calculated for the given scenario (for example by using the previous calculation script). It loads all Category Combinations, but sets the chosen combination as the ‘Headline result’.

The Receiver area is input parameter 1, scenario name is parameter 2, Combo is parameter 3.

To use this script, connect to the required database and type (substituting the correct area, scenario and combination names):

```
run loadresults  "Area" "Scenario" "Combo"
```

Example of command line, showing use of quotes:

```
run "\\Cerver\Script files\loadresults.nms" "Burton Estate" &
"With New Road" "All Roads"
```

SHOWRECCHANGE.NMS

This script calculates the change of receiver level between two scenarios. It assumes the user has connected to the required database. (It does not matter if this is a local or remote database.)

The Calculation area is input parameter 1, the first scenario name is parameter 2, the second scenario name is parameter 3. The Category combination for the comparison is parameter 4.

To use this script, connect to the required database and type (substituting the correct area, scenario and combination names):

```
run ShowRecChange "Area" "Scenario 1" &
"Scenario 2" "Combo"
```

Example of command line, showing use of quotes:

```
"\\Cerver\Script files\showrecchange.nms" "Burton Estate" &
"Base Scenario" "With New Road" "All Roads"
```

EXPLANATION OF THIS SCRIPT

This script downloads some receiver results and then uses the Results Processor to calculate the change of noise level. This uses some advanced features which are described here.

Displaying receiver results is less straightforward than contours because there are more steps involved due to the fact that all the noise indexes and combinations are stored together: they must be downloaded together and separated later.

Before using the script, you must have connected to the required database.

Firstly, the script **clears** the model which ensures we start with a clean slate. Then it sends a **message** to the user so that they know the script has started. It confirms the parameters that will be used. (The presumption is that the second-named scenario is the 'future' and the first-named scenario is the 'base', so the script will show the change of level from the base to future situation.)

Then the **load model** command loads the model covering the required area for the second-named scenario. The graphical display will show this area automatically.

Next, the **Load Results** command loads the receiver noise levels for this area, for the L10 method and for the first-named scenario. This will be known in NoiseMap as Resultset number 1.

Next, the **Load Results** command loads the receiver noise levels for this area, for the L10 method and for the second-named scenario. This will be known in NoiseMap as Resultset number 2.

Load Results downloads all the combos that have been calculated for this selection. You now need to select the combo and sub-set of calculations from the ResultSet.

The **add receiverresult** command selects the combo from the resultset. This creates a result which can be named. In this case we chose R1 and R2 respectively, as it is easier to enter these into the Results expression in the next step, but you can use other names if you wish. This step selects but does not display the result.

We now enter the result expression that we wish to display using **display results**. The expression is R2-R1 which is the change of level from the first scenario to the second scenario in this case, i.e. from the Base to the Future.

However, beware that although you can now download this result, it does not actually show the result on-screen – this needs a further step.

You can use the **Label** options or **View-As-Colour** from the Graphical Interface.

View-As-Colour

This does not have a script command. Select View-As-Colour from the menu, then choose Receivers; Results and then choose Set Result. You will note that 'Change of Level' is now one of the options.

SHOWCONTCHANGE.NMS

This script shows a contour of the change of level between scenarios. It assumes user is already connected to the required database, which can be a remote or local database.

The Calculation area is input parameter 1, first scenario name is parameter 2, second scenario name is parameter 3, Category combination is parameter 4.

This script shows how to set up the display parameters for a contour.

To use this script, type:

```
run ShowContChange  "Area" "Scenario 1" &
"Scenario 2" "Combo"
```

Example of command line, showing use of quotes:

```
run "\\Cerver\Script files\ShowContChange.nms" "Burton Estate" &
"Base Scenario" "With New Road" "All Roads"
```

TESTFIND

This script shows how to use the Test script command to test the script operation of NoiseMap's Find function. The find function is implemented by the script command `Search`.

This script shows how the various search commands are coded. The test is made on the `manualexampledatabase.nmdb` model.

Some of the expected values have not been updated and will fail, but the script shows how the various search options are implemented.

SCROLL THROUGH 24-H CONTOURS

This script sequences through 24 1-hour contours, showing how the noise level varies throughout the day, looping round 10 times. It uses contours already calculated and stored in a database `noisemap24scroll.nmdb`.

It may be noted that because arithmetic operations are not allowed on script variables, it is not possible to use them as a variable parameter in this case.