Logicals, command aliases, and hotkeys

Creating a macro script to automate some process in Surpac is an easy task to perform. However, having this macro work consistently from any directory can be difficult. Using logicals when you develop macros is crucial to achieving portability across directory structures. The use of command aliases and hotkeys then provides you with a mechanism to quickly access and run common macros.

Logicals

'Logical' is a Surpac term that refers to the use of a known name to map to a physical directory. The known name, or 'logical', is mapped to a directory on your hard disk at runtime. Using this system your macros do not have to refer to a hard coded directory, which might be different on every installation of Surpac, to access files.

For example, the logical name ***SSI_ETC:*** (the known name) can be mapped to a directory named **c:/users/Public/Gemcom/Surpac/63/share/etc** on one system or **s:/software/Surpac/v6/share/etc** on another. Surpac uses the known name ***SSI_ETC:*** to refer to the directory, this means it does not need to know where this directory actually exists because it is mapped at runtime.

The reasons logicals are used are:

- to insulate the software from the file system
- to shorten notation for accessing long directory paths
- to standardise data

Logicals come in three types:

- ***System*** – defined in the **translate.ssi** file
- ***User*** – define in the **SSI_ETC:logicals.ssi** file
- ***Personal*** – defined in any file and specified to *Surpac* using the **Customise > Default Preferences** function

System logicals are defined by the software installation procedure and are stored into a file that is usually called **translate.ssi.** The system logicals are mandatory. You should never modify a system logical. Below is an example **translate.ssi** file.

```
MACHINE=WIN32
SSI_ETC:          c:\Documents and Settings\All Users\Gemcom\Surpac\61\share\etc\
SSI_STYLES:       c:\Documents and Settings\All Users\Gemcom\Surpac\61\share\styles\
SSI_PLOTTING:     c:\Documents and Settings\All Users\Gemcom\Surpac\61\share\plotting\
SSI_PROFILES:     c:\Documents and Settings\All Users\Gemcom\Surpac\61\share\profiles\
SSI_HMF:          c:\Program Files\Gemcom\Surpac\61\share\hmf\
SSI_MESSAGES:     c:\Program Files\Gemcom\Surpac\61\share\msg\
SSI_REFMAN:       c:\Program Files\Gemcom\Surpac\61\share\refman\
SSI_RESOURCE:     c:\Documents and Settings\All Users\Gemcom\Surpac\61\share\resource\
SSI_JAVA:         c:\Program Files\Gemcom\Surpac\61\share\java\
SSI_BIN:          c:\Program Files\Gemcom\Surpac\61\nt_i386\bin\
SSI_LIB:          c:\Program Files\Gemcom\Surpac\61\nt_i386\lib\
END
```

Caution: Do not add user logicals to the translate file. This is because at each new installation of the software this file is created and so you will lose any changes that you had made.

User logicals are optional and you can use them to make finding your macros and data easier. In the same way that system logicals make it easy for Surpac to locate files, user logicals make it easy for you to locate your data and macros. For example, you could define logicals to centralise the storage of your survey pickups, or make a repository used to store all macros so that each user on a site can access them using a consistent name. Using a logical to access macros and data using a standard naming convention, where the physical location of files on a disk does not matter, can be very helpful when you are working in a network environment, especially when users have different physical drive mappings.

User logicals are also a valuable tool when you are designing menu systems and macros. By using logicals, the menu definitions and macros can be kept insulated from the file system because there is no need to worry about physical drive mappings. This way when menus and scripts are transferred to other computers, it is only the logical reference in the **logicals.ssi** file that needs to be changed and not the actual script or menu definitions.

User logicals are defined in a file called **logicals.ssi** that is stored in the ***SSI_ETC:*** directory. This is a system logical that maps to the directory where the etc files are stored. The **logicals.ssi** file is optional, if this file does not exist is does not cause an error in Surpac. However, if it does exist then Surpac will load the file and add any defined logicals to the system logical map.

Personal logicals are similar to user logicals except that they can be defined in a file with any name. For example, you can store personal logicals in a file called **mylogicals.txt** .To make Surpac load a personal logicals file you must define it in Surpac using the **alias tab** of the *Settings* form. You can access this form by choosing **Customise > Default preferences**. The format of a personal alias file is exactly the same as the **logicals.ssi** file.

When Surpac starts, it will load logical definitions in the following order:

1. System logicals from **SSI_ETC:translate.ssi**.
2. User logicals from **SSI_ETC:logicals.ssi**.
3. Personal logicals from your defined logical file.

If there are duplicate logical definitions then the last one read is the one that takes precedence. It is not an error to define duplicate logicals and you might have good reasons to do so, such as redefining the location of the **SSI_PLOTTING:** logical in your personal logicals file.

Below is an example of a **logicals.ssi** file or a personal logicals file.

```
MINESOLUTIONS: c:/minesolutions/
MS_SPOOLER:    c:/minesolutions/spooler/
MS_STRING:     c:/minesolutions/string/
MS_RINGKING:   c:/minesolutions/ringking/
MY_MACROS:     c:/macros/
```

The structure of the file is quite basic. You first define the logical name, leave some white space, this can be one or more spaces, then define the physical directory mapping followed by a trailing slash.

Note: The trailing slash is mandatory, but it can be either a forward or backslash.

Tips:

- Use the full colon at the end of each logical. It provides a consistent end-of-logical marker and makes file paths easier to read when using logicals.
- Forward slashes are recommended over backslashes for your physical paths. Backslashes can sometimes cause obscure problems due to their use as an escaping function in Tcl scripts.
- Make sure your physical directory names are correct; Surpac does not check to see if they actually exist.

Task: Create a user logical

1. Open ConTEXT.
2. Choose **File > New**.
3. Determine the path name of your current working directory in Surpac.

Note: The pathname of the current work directory is displayed in the title bar of the Surpac window.

4. Define a logical called MY_WORK: that will map to the current work directory. Your definition will look similar to the following extract from the **default.ssi** file:

```
MY_WORK: c:/Documents and Settings/All Users/Gemcom/Surpac/61/demo_data/tutorials/tcl_scl/
```

5. Choose **File > Save As.**
6. In the *Save As* form, navigate to the SSI_ETC: directory, and type **logicals.ssi** for the file name, then click **Save**.
7. If you are running Surpac, exit and then restart.
8. Locate your new logical called MY_WORK: in the Navigator.

   Note: All logical names are listed in the Surpac Navigator beneath the folder names in alphabetical order.

Tip: If you cannot find your MY_WORK: logical in the list check that you have named the logicals file correctly as described in step six. Make sure you know what the actual pathname for SSI_ETC is on your computer. You can identify this pathname by finding the SSI_ETC logical in the Surpac Navigator and expanding it.

Command alias

A command alias is a system that provides a mechanism for you to rename commands in Surpac to something that you find more suitable or easier to remember. It is often quicker for experienced Surpac users to make use of these short cut alias names than it is to type the full command name or find the function on a menu or toolbar.

Command aliases are defined in a text file with a set syntax. First, the alias name is given, enclosed in double quotes, followed by any amount of white space, and then the actual command name, enclosed in double quotes. An extract from the distributed **short.mst** alias file follows:

```
"2DG"       "2D GRID"
"2DT"       "2D TRANSFORM"
"3DG"       "3D GRID"
"3DT"       "3D TRANSFORMATION"
"ATP"       "ADD TO PERIOD"
"AR"        "ADD RIG"
"AB"        "APPLY BOUNDARY"
"AII"       "ARC ARC INTERSECT"
"AD"        "AUDIT DATABASE"
"BS"        "BASIC STATISTICS"
"BD"        "BEARING AND DISTANCE"
"BR"        "BENCH REPORT"
```

Tip: When defining alias names that clash, the last one Surpac reads is the one that takes precedence. There is no warning message for a duplicate name so be careful when writing your alias file.

As well as allowing you to rename existing commands, the alias system also allows you to define new command names that you can associate with your Tcl/Scl scripts. For example, if you have defined two scripts, one to import a csv file and load into a database table, and another to export a database table to a csv file, you can create keyboard commands that will run these scripts. The following alias file defines two new commands **import_csv** and **export_csv** to run macros stored in the **MY_WORK:** logical.

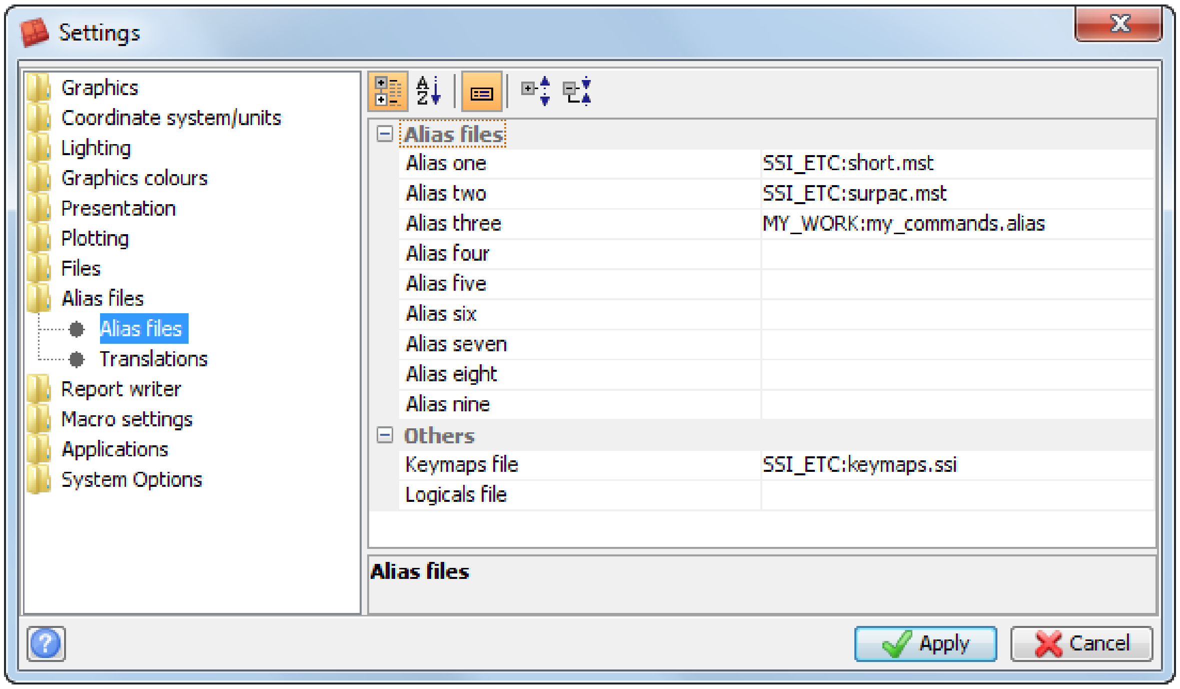
```
"IMPORT_CSV"      "MACRO:MY_WORK:IMPORT_CSV"
"EXPORT_CSV"      "MACRO:MY_WORK:EXPORT_CSV"
```

Later you will learn how to attach these two scripts to menus.

Note: The keyword **MACRO:** is used in the path above. This keyword tells Surpac that it is not mapping to an internal Surpac command but instead is going to run a Tcl script that is stored on the hard disk. Notice the use of the logical **MY_WORK:** which will map to the actual directory path where the macro is stored. This logical is assumed to be previously defined in **logicals.ssi**.

Surpac allows you to specify up to nine alias files.

To define an alias file in Surpac, choose **Customise > Default Preference,** select the **Alias files** folder and enter the name of your alias file including any logical or physical directory name.



Alias files	
Alias one	SSI_ETC:short.mst
Alias two	SSI_ETC:surpac.mst
Alias three	MY_WORK:my_commands.alias
Alias four	
Alias five	
Alias six	
Alias seven	
Alias eight	
Alias nine	
Others	
Keymaps file	SSI_ETC:keymaps.ssi
Logicals file	

Alias files

Task: Create a command alias

1. Open ConTEXT.
2. Choose **File > New**.
3. Type the following line.

```
puts "Hello Surpac - this is my first script"
```

Note: Make sure you type exactly as shown, taking care with upper and lower case letters.

4. Choose **File > Save As**.
5. Navigate to the current work directory, and type **hello.tcl** for the file name, then click **Save**.
6. Choose **File > New**.
7. Now create a new command alias definition called HELLO that will run the script created in steps 2 and 3. Your alias definition in the editor should look as follows:

```
"HELLO"           "MACRO:MY_WORK:hello.tcl"
```

Note: The **MY_WORK:** logical is the logical name created in the previous task. Using the logical name as part of the alias definition means that the command will work from any directory. If you do not use a logical the macro will work only if the current work directory is the one the macro is stored in.

8. Choose **File > Save As**.
9. Leave the location as the current work directory, and type **myCommands.txt** for the file name, then click **Save**.

10. In Surpac, choose **Customise > Default preferences**. Then enter the full pathname to the alias file you have created in steps 6 to 9.
11. Exit Surpac and then restart.
12. In the Function choose, type ***HELLO***.

    You should see the following output in the message window:

    ```
    Hello Surpac - this is my first script
    ```

    Tip: When defining alias commands to run scripts it is safer to make use of logicals as in the example above.

Keymaps

The keymaps file is a historical file used in earlier versions of Surpac to map the keyboard keys to characters. In Surpac, the only real purpose of the file is to define actions associated with the function keys. You can use the keymaps file to define your own hotkeys to run scripts that you have written.

Using the csv example from the previous discussion of the alias, the following **keymaps.ssi** file extract shows how to define hot keys on the F7 and F8 keys to run the import/export csv scripts.

```
"f7"          FUNCTION    "MACRO:MY_WORK:IMPORT_CSV"
"f8"          FUNCTION    "MACRO:MY_WORK:EXPORT_CSV"
```

Tip: When defining hot keys the key name must be in lowercase characters (for example, f10 not F10).

Task: Create a hotkey

Note: This task assumes you performed the **Create a command alias** task in the previous section.

1. Open ConTEXT.
2. Choose **File > Open**, and open **SSI_ETC:keymaps.ssi**.
3. Scroll down and locate the section that defines the F1 through F10 keys.
4. Define a hotkey F11 to run the **hello.tcl** script by inserting a new line as follows:

    ```
    "f11"      FUNCTION       "MACRO:MY_WORK:hello.tcl"
    ```

    Note: The **MY_WORK:** section is the logical name created in a previous task. Using the logical name as part of the hotkey definition means that the command will work from any directory. Otherwise the hotkey would work only if the macro is saved in the current work directory.

4. If you are running Surpac, exit and then restart.
5. Press **F11**.

    You should see the following output in the **message window**:

    ```
    Hello Surpac - this is my first script
    ```