

# General Information on Script Files

This chapter covers the following topics:

- What is a Script File?
  - Script File Folders
  - Startup Scripts
  - Using Backslashes
  - Example Script File
  - Running Script Files in the Development Environment
- 

## What is a Script File?

Script files (extension *js*) initiate operations which are executed on the Entire Screen Builder Server. You can create script files with an ASCII editor on your PC. For example, you can create a script file which automates the process of logging on to a host system. All required logon methods are contained in the script file and are executed when the script file is invoked.

### **Important:**

The extension *js* must be in lower-case when using the Entire Screen Builder Server on UNIX. We also recommend that you use lower-case for the script file name.

Scripts files can be used by all types of viewers (Windows Viewer, Web Viewer and Terminal Viewer).

## Script File Folders

The script files are stored in a script files folder on the same machine on which the Entire Screen Builder Server is located. For a default installation, the script files folder is `\Entire Screen Builder 5\scripts`.

You can change the script files folder using the System Management Hub. See *Server Settings* in Entire Screen Builder's *System Management Hub* documentation.

The script files folder contains two subfolders: *production* and *test*. For normal operation, the script files must be stored in the *production* subfolder. If you want to test and debug script files, these files must be stored in the *test* subfolder. Note that debugging is only possible with the Terminal Viewer.

Sample script files are copied to your hard disk during installation. They can be found in the folder `\Entire Screen Builder 5\samples\samplescript`.

## Startup Scripts

A startup script is executed after a user has logged on to the Entire Screen Builder Server. It is defined in the user profile using the System Management Hub. See *Users* in Entire Screen Builder's *System Management Hub* documentation.

- **GUI Viewers**

A startup script for a user can only be invoked when anonymous logon has been disabled and when the connection number 0 has been specified. A detailed description for these attributes can be found in the *Overview of Client Control Properties* which is part of the *User Exits* documentation.

- **Terminal Viewer**

The startup script is executed immediately after the user has logged on to the Entire Screen Builder Server.

Calling the startup script does not establish a connection between the viewer and the Entire Screen Builder Server. The connection to the server is only established after a successful connection to a host session using the communication methods `OpenSession` or `OpenUpdatedSession`. Due to this restriction, several script methods (which are partially executed in the viewer) can only be executed after a connection to a session has been established. Otherwise, `NSW.errno` is set to `NSW_NO_OPEN_SESSION`. This applies to the user interface methods `MessageBox`, `OutputMsg`, `GetUserInput` and all client-side access methods.

## Using Backslashes

As with most programming languages, the backslash has the special meaning of an escape character. Together with the character by which it is followed, the backslash defines a special character constant (for example, `"\n"` means "new line").

If the backslash is part of a string, you have to double it. Example:

```
Result = NSW.SetDataTransfer("SET PCFILE 7 DOWN DATA C:\\\\TEMP\\TEST.NCD");
```

You can also use one forward slash instead of two backslashes. Thus, the following statements are both valid:

```
dwErrorReturn = NSW.CreateClientFolder ("C:\\Myfolder")  
dwErrorReturn = NSW.CreateClientFolder ("C:/Myfolder")
```

## Example Script File

The following example shows how to logon to a VM mainframe system:

```
// logon.js: sample script file to logon to a mainframe system

if (NSW.GetScreenText(18,8,7) == "NETwork")
{
NSW.TypeString(-1, -1, "demouser", TAB);
Passwd = NSW.GetUserInput("Password-Request", "Please enter your password", PASSWORD_CHAR);
NSW.TypeString(-1, -1, Passwd, CR);
}

bWait = true;

while (bWait == true)
{
if (NSW.GetScreenText(5,3,5) != "Level")
    NSW.Pause(1000);
else
bWait = false;
}

NSW.SendKey(-1, -1, PF1);
NSW.Pause(2000);
if (NSW.GetScreenText(1,30,17) != "C O N - N E C T 3")
NSW.OutputMsg("Logon failed!");
```

## Running Script Files in the Development Environment

If a script file is being run from the SDK, sufficient time should be allowed for the screen to update between commands.

For example, if the script contains the following code

```
for (loop = 0; loop < 10; loop++)
{
    NSW.SendKey(-1, -1, PF3);
}
```

you should add a Pause statement to allow for the extra screen processing required in the development version:

```
for (loop = 0; loop < 10; loop++)
{
    NSW.SendKey(-1, -1, PF3);
    NSW.Pause(1500);
}
```