Using the Microsoft Script Editor to Debug DIAdem User Dialog Boxes (SUD) and DIAdem DataPlugins

General Information

You can debug DIAdem user dialog boxes and DIAdem DataPlugins with the Microsoft script editor. This document describes which settings you must make on your computer to use the Microsoft script editor.

To debug a script, the script must be unencrypted. This is why you **cannot** debug user dialog boxes that have the filename extension .Suc.

Note:
Debugging DIAdem user dialog boxes (SUD) and DIAdem DataPlugins is an "unsupported feature".

Warning:
To debug DIAdem user dialog boxes and DIAdem DataPlugins, you might have to make changes to the Windows registry. Only experienced users should make changes to the Windows registry. Errors might destabilize your system.

Requirements

For debugging, you need **DIAdem** version 9.1 and later, **Microsoft Windows Script,** and a suitable VBS script debugger.

We recommend the **Microsoft script editor (version 10 or later)** from the **Microsoft Development Environment (version 7.0 or later)** as a VBS script debugger**.** Microsoft usually installs these components with the Microsoft Office products if they support VBA. You can add this feature later if you do not have it on your computer. Refer to the help section for Microsoft Office products for further information.

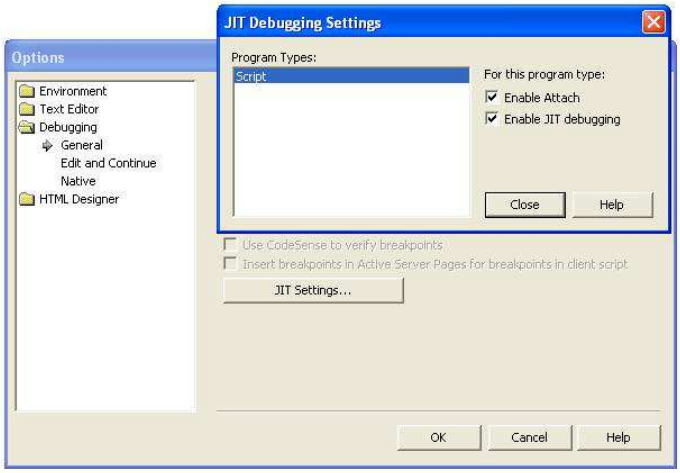
All pictures and descriptions in this document were created with the **Microsoft script editor version 10 (Microsoft Development Environment 7.0).** They might be different in other versions.

Tip:
Start the Microsoft Script Debugger from Microsoft Word or any other Office application, to check whether the Microsoft Development Environment is installed correctly. You can install missing components automatically.

Tip:
The Microsoft script editor is on your hard disk under MSE.exe or MSE7.exe.

Settings

Enabling JUST IN TIME (JIT) Debugging in the Microsoft Script Editor



To debug scripts, you must first activate JIT debugging. Select **Extras»Options** and set **Enable attach** and **Enable JIT debugging** to enable JIT debugging in the Microsoft script editor.

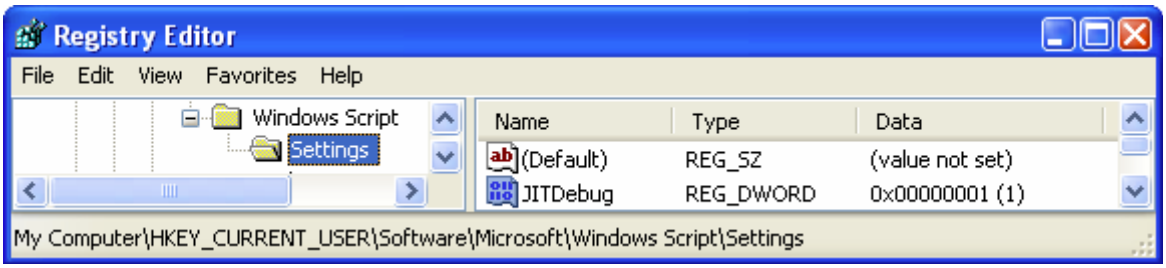
Enabling VBS Debugging for the Whole System

In Microsoft VBScript Version 5.1 and later versions this feature is not enabled. Refer to http://support.microsoft.com/default.aspx?scid=kb;EN-US;252895) for further information.
To enable this feature you must check the Windows registry and you might have to create a new key.

You can find the **JITDebug** key in the Windows registry on one of the following paths:

HKEY_CURRENT_USER\Software\Microsoft\Windows Script\Settings
or
HKEY_USERS\.DEFAULT\Software\Microsoft\Windows Script\Settings

To start the Microsoft registry editor „Regedit.exe“ and change the Windows registry, select **Start»Execute** in Windows, enter Regedit.exe, and click **OK**.
The **JITDebug** key is a DWORD entry. If the value is 1, the JIT is enabled. If the value is 0, the JIT is disabled. Ensure that the value is 1. Generate the path and the key manually if they do not already exist.



Note:
***This change in the Windows registry is a system-wide setting. If script errors occur in other programs, for example, in the Microsoft Internet Explorer, a query automatically asks whether you want to debug the program. Disable this functionality by resetting the JITDebug value to 0*.**

Tip:
You can export the key if you do not want to open the Windows registry whenever you reset the switch. *Refer to the help section for the Microsoft Windows registry editor for information on how to export the registry entries. First export the key with the value 0, then the key with the value 1. Double-click the file to enable or disable this feature.*
Debugging Options

Defining Breakpoints

Use the **Stop** command to interrupt a script at a specified point. The script editor opens automatically and you can debug the script starting from this point.

Example:

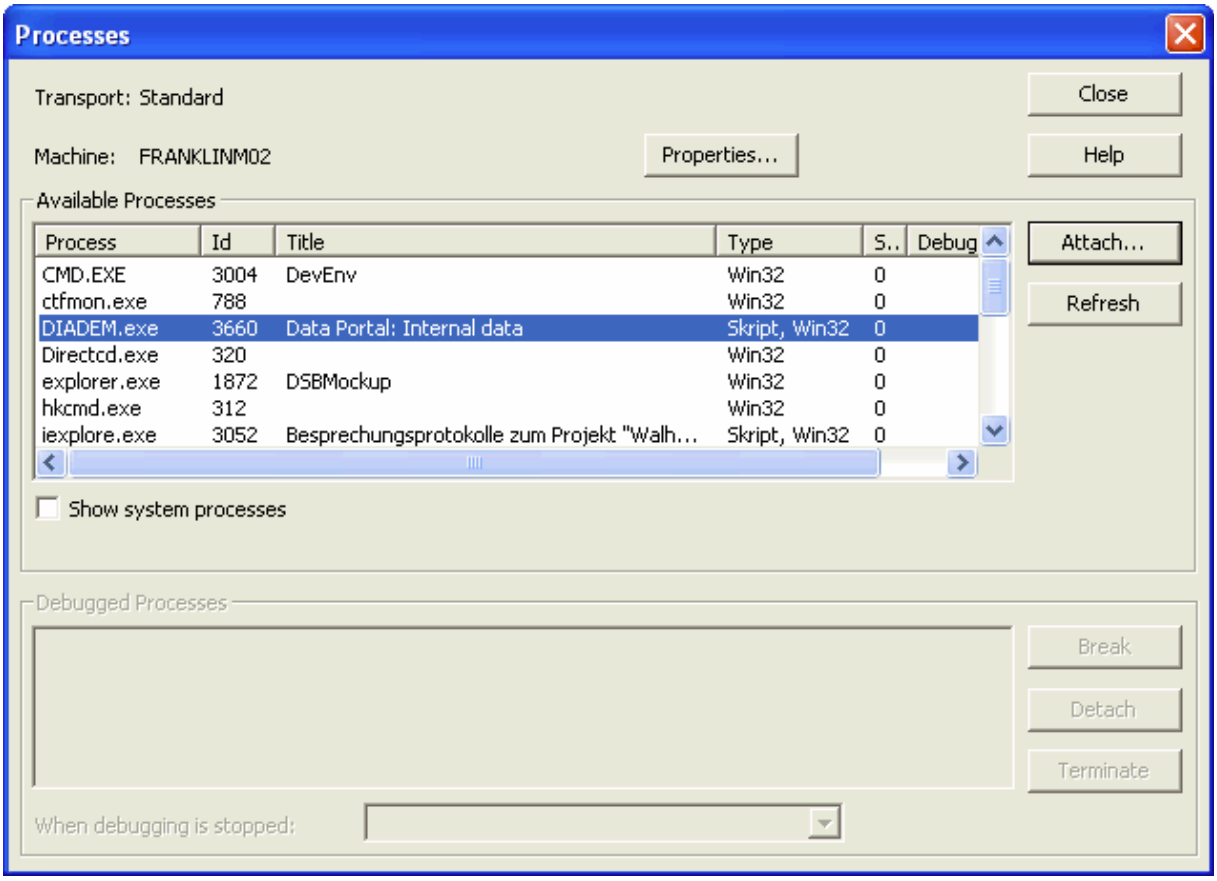
```
Sub Foo()
        Dim i
        Stop
        i = 1+1
End Sub
```

Note: The **Stop** command is only executed if system-wide VBS debugging is enabled.

Note:
Remove all „Stop“ instructions before you pass on your code to customers.

Debugging an Active Script

Start the Microsoft script editor and select **Debugging»Processes.** The following dialog box appears:



Note:
*To open the **Processes** dialog box, you must start the Microsoft script editor in the Explorer. If you start the Microsoft script editor in a Microsoft Office product, you might **not** be able to use the option of adding a process to a running process.*

The **Processes** dialog box displays the enabled processes**.** If the **Type** column displays „Win32" in the **DIAdem.exe** line, you cannot add to a process. In this case check the settings from the above section.
If the **Type** column displays „Win32, Script" in the **DIAdem.exe** line, click the „Add" button, select **Script** in the dialog box that opens, and click **OK.** Close all dialog boxes.

The Microsoft script editor displays DIAdem as an active document. You can use the debugging toolbar to enable the window that contains the active documents.

If you run a script in DIAdem in a user dialog box or in a DataPlugin, this dialog box or DataPlugin appears as an active document in the window that shows the active documents.

Note:
The Microsoft script editor displays DIAdem user dialog boxes or DIAdem DataPlugins only. You only can debug all other DIAdem scripts with the internal DIAdem debugger. If you double-click the active document, the editor displays the source code of the active document. You now can set breakpoints.

Debugging When an Error Occurs

If JIT is activated on your computer, a dialog box automatically opens when an error occurs in a script. You can select the Microsoft script editor in this dialog box.

Using the DBM Command

Use the **DBM** command to output texts in a debug window. As of DIAdem 2014, this command is available in DIAdem scripts, DataPlugin scripts, and scripts in dialog boxes. In order to see the debugger messages, you need an appropriate program, for example, DebugView (http://technet.microsoft.com/en-us/sysinternals/bb896647.aspx).

Example:

```
Sub Foo(InputPar)
        Call DBM("Input parameter: " & InputPar)
End Sub
```