

A POWERPRO PLUGIN TO MANAGE DDE

Alan Campbell

No warranty of any kind, express or implied, is included with this software; use at your own risk Responsibility for damages (if any) to anyone resulting from the use of this software rests entirely with the user.

1.0 Overview

This PowerPro plugin allows you to communicate with DDE servers. DDE is rather old fashioned, but has its advantages. See

<http://www.angelfire.com/biz/rhaminisys/ddeinfo.html>

Applications that support DDE include: Opera, Netscape, Internet Explorer, the MS Office applications, Matlab, ObjectVision, Paradox, WordPerfect, Quicken. It's also quite a popular way to interface to data collection devices. It can be made to work with remote servers.

1.1 This Document

There are three versions of this document, with the same content. There's an RTF file, which looks nice in Word but is something like 150k in size; there's a compiled help (chm) document, which is much smaller if somewhat uglier; and there's a pdf, with bookmarks for each section heading.

In my experiments I've found the RTF file doesn't display correctly in anything but Word (not Keynote, not even Wordpad: you've think Microsoft could at least get their rtf engines consistent). So if you don't have Word, better use the chm file.

All documents have extensive hyperlinks. The table of contents at the front of the rtf and chm documents is a set of them.

The chm file has no index.

1.2 What's New In This Version

- fixed bug in start_conversation that permanently prevents further conversations if start_conversation fails.

2.0 Requirements

Powerpro version 3.4 or later. Test scripts require at least 3.8.15. Edit them to get rid of ?c...c syntax to use with previous versions. Works in Standard Configuration.

Those of you who are tetchy about MFC will be happy to know that, unlike the registry plugins, dde.dll *doesn't* require MFC support (e.g. MFC42.DLL).

2.1 Related plugins

If the server you want to use supports COM, the com plugin might be easier to use.

If your external program offers an API, you could use the dll plugin to get at it.

2.2 Reporting Bugs, Requesting Enhancements

If you hit any problems with this plugin (or any of my plugins, for that matter), it'd be helpful if you reported them via the PowerPro forum (<http://groups.yahoo.com/group/power-pro/>) in a message with a clear subject line (maybe: "DDE PLUGIN: apparent error in:..."). I don't read everything in the forum, but I will see anything flagged with an obvious header. Please include a copy of the script causing problems, and state which version of PowerPro and of the dde plugin you're using.

3.0 File list

plugins\dde.dll

doc\ddePluginReadme.rtf
doc\ddePluginReadme.chm
doc\ddePluginReadme.pdf
doc\ddePluginFunctions.txt

scripts\ddePluginTestScriptExcel.powerpro
scripts\test.xls

scripts\ddePluginTestScriptPdf.powerpro
scripts\test.pdf

scripts\ddePluginTestScriptEccoPro.powerpro

scripts\ddePluginTestScriptOutputProvided.log

scripts\dde.ini

4.0 Installation

Copy dde.dll from ddePluginX.XX.zip archive into your PowerPro directory, or into its Plugins subfolder. If you want to provide an initial configuration of the plugin (see Section 9.1), edit dde.ini and put it in the folder pointed to by pprofolder; or add its edited contents to plugins.ini in the same folder.

The .powerpro scripts and .log sample output can go wherever you want. If you want the test script ddePluginTestScriptExcel.powerpro to work correctly, keep it in the same folder as test.xls; and test.pdf needs to be with ddePluginTestScriptPdf.powerpro.

If you want [dde.help](#) to automatically open the chm/rtf documentation, you may want to put one of the help files in the plugins and maybe rename them. See section [10.13](#).

ddePluginFunctions.txt can be appended to pprofunctions.txt supplied in the PowerPro distro (or included in it, using

```
include <path to>\ddePluginFunctions.txt )
```

It can then be accessed either as part of pprofunctions.txt or on its own as a file menu (e.g. using a hot key associated with

```
*keys {filemenu <path to>\pproFunctions.txt} )
```

All other files can go wherever you want. But if you want the test script ddePluginTestScript.powerpro to work correctly, it should be in the same folder as testPDF.pdf.

dde.dll is about 31k in size. If you want, you can use upx (<http://upx.sourceforge.net/>) to compress it down, but not really sure it'd be worth it. Read about the pros and cons at <http://en.wikipedia.org/wiki/UPX>.

5.0 Uninstall

Remove all files listed in the above section (“[3.0 File list](#)”) from your PowerPro folder.

6.0 Acknowledgements

Thanks to Randall Stukey and Sheri Pierce for catching unnecessary length limitations in services.

7.0 Testing

If you want to use the enclosed scripts, “ddePluginTestScript*.powerpro” to exercise the plugin:

Either:

- Put a ddePluginTestScriptXXX.powerpro in your <PowerPro configuration>\scripts directory
- Run a ddePluginTestScriptXXX.powerpro with a command (menu item?) in PowerPro like

**Script RunFile ddePluginTestScriptXXX.powerpro or
.ddePluginTestScriptXXX*

Or:

- Put ddePluginTestScript*.powerpro scripts anywhere and double click on it (as long as PowerPro is already running). This didn't work on my machine until I manually set up an association with .powerpro, but if your Powerpro installation went correctly it should work for you.

The ddePlugin*.powerpro scripts will output to the Debug window, and, if you uncomment the line

```
;global g_fh = file.open(dde_test_log_file, "a")
```

in each script, to a log file. By default that file is "ddePluginTestOutput.log" in the same directory as your pcf file. You can edit the script to point it another path/file if you wish: it's the first line of code in the script, and assigns a path/file name to **dde_test_log_file**.

The ddePluginTest*.powerpro scripts don't use the evaluate-expression operator "&" so is not dependent on your choice for it. It uses the ?c...c syntax to avoid problems with your declared escape character, so you should have no problems whether that's ' or \.

The log file generated by the ddePluginTestScript*.powerpro scripts can be compared with "ddePluginTestScriptOutputProvided.log" that comes with the distribution.

There are further comments on how each ddePluginTestScript*.powerpro script works embedded in the script itself.

Notes on specific scripts:

ddePluginTestScriptExcel.powerpro: Requires Excel, probably almost any version. Manipulates test.xls, which must be in the same folder as the script. Opens the file, runs a command, pokes some data, requests some data.

For details on using DDE with Excel see

<http://www.angelfire.com/biz/rhaminisys/ddeapps.html>
<http://www.angelfire.com/biz/rhaminisys/ddeinfo.html>

ddePluginTestScriptPdf.powerpro: Requires Acrobat Reader, or Acrobat, probably almost any version. Manipulates test.pdf, which must be in the same folder as the script. Opens the file, executes some commands. Uses the reg plugin to find out where to get Acrobat Reader.

The example code came from SeaSideTech (www.seasidetech.net). Full details on Acrobat and DDE (as well as COM) can be found in "Acrobat Interapplication Communication Reference" at

<http://partners.adobe.com/public/developer/en/acrobat/sdk/IACReference.pdf>

ddePluginTestScriptEccoPro.powerpro: Requires EccoPro, available from

http://www.tuleriversoftware.com/downloads/EccoPro_4.01_UpdatedInstall.exe

8.0 List of Services And General Notes on Usage

Ensure dde.dll is in your PowerPro installation directory, or in the plugins subfolder thereof.

There are numerous services in this plugin. They are

service	description	section
start_conversation	creates a connection to a server, i.e. an application/topic pair	10.1
stop_conversation	disconnects from a server	10.2
request	request data from server	10.3
execute	start transaction to execute a command on server	10.4
poke	start transaction to poke data at server	10.5
advise	start transaction to start monitoring an item on the server	10.6
get_last_result	get last data delivered by advise loop	10.7
advise_stop	start transaction to stop monitoring an item on the server	10.8
check_service_complete	determine if transaction complete	10.9
error_dialog_on error_dialog_off	turn PowerPro script error dialog on and off	10.10
version		10.11
help		10.12
config	set location of configuration ini file	10.13
unload	remove plugin from memory	

These are described below, in [Section 10](#) and its subsections.

9.0 Writing scripts using the dde plugin

Call the appropriate plugin service as follows:

```
retval = dde.<service>(arguments if any)
```

Results, if any, are generally available as the returned value from the above expression, though there are alternatives.

After you are finished using the dde plugin in your script, you can if you wish unload it with

```
dde.unload
```

It's probably best done if you do not foresee the plugin being used again for a while.

If you unload the plugin, it's behaviour returns to the default in any subsequent call, i.e. further dde service calls will return values.

There are other services which affect the behaviour of the dde plugin (see sections [10.10](#), [10.13](#)). You can also customise the behaviour of plugin services by providing a configuration ini file (see section [9.1](#)).

If you do an *Exec ChangeConfiguration, and the new and old pcf files are in different folders, you should unload this plugin before using any of its services with the new configuration. (only necessary if you use relative paths to specify the location of ini files).

There are a few icons embedded in dde.dll. You could use them as PowerPro list item icons; or as an dialog icon if you're using the dialog plugin; or as an image in such a dialog.

9.1 The Configuration ini File

On the first call to any dde service, the plugin checks for a file called dde.ini in the folder pointed to by the Powerpro variable pprofolder (usually the folder in which the currently active pcf file is found). It then looks for a [ddeConfig] section in that ini file. If that's not found, it looks for the same section in the file plugins.ini in the same folder. If either are found, it looks for the following values in the section:

key	possible values	default value	can also change using service	meaning
raiseErrors	0 1 y n t f*	1	error_dialog_on error_dialog_off	determines if errors in syntax format, service arguments etc cause powerpro to raise the script error dialog.
insertionMarker	any string < 5 chars long	#	none	sets string used to mark point of insertion for results in result commands for threaded threaded services
helpFileLocation	path	null	none	set to location of chm or rtf help file

* Only the first non-whitespace character of the key's value is checked, no matter how many there are. So values "yes", "no", "true" or "false" will work. As usual the ini value is case-insensitive, so "Y" and "FALSE" are also valid.

† Only the first non-whitespace character of the key's value is checked, so "status", "data" and "none" will work.

If no dde.ini or plugin.ini files are found with a [ddeConfig] section, the dde plugin will initially just use the compiled-in, default values (specified in the third column above) to configure itself.

If an ini file with a [ddeConfig] section is processed, and there's a possible value that could be in ini but isn't, the dde plugin reverts to the default value for that value.

Once an ini file with a [ddeConfig] section is processed, its values stay in force until the dde plugin unloaded or the config service is run.

If the first service called is one which itself changes the configuration (e.g. **error_dialog_on**), the configuration ini file will be found and evaluated *before* the service is applied.

If you want to change your initial configuration, you can use the dde plugin itself to do it. E.g.:

```
ini.set(pprofolder ++ "plugins.ini", "ddeConfig", "missingKeysError", "no")
```

After making your changes, either:

- unload the dde plugin. The next time you use one of its services, the new configuration will kick in. Or
- run **dde.config(<path to configuration ini file>)**.

If you do an *Exec ChangeConfiguration; and if the new and old pcf files are in different folders; and if there's a dde.ini or plugins.ini file in the new pprofolder, the configuration it specifies won't take effect until you take one of the steps above..

10.0 The Services

Details of specific services follow.

The next sections describe services that affect the behaviour of all others.

10.1 start_conversation

dde.start_conversation(application, topic)

Creates a connection to a server, i.e. an **application/topic** pair

Usually the application in question has to be running already. That's DDE for you.

application and **topic** must be less than 255 characters in length (a DDE limitation).

See Section [11.2](#) "Values Returned by DDE-Related Services" for what the service returns.

10.2 stop_conversation

dde.stop_conversation()

Disconnects from a server.

See Section [11.2](#) "Values Returned by DDE-Related Services" for what the service returns.

dde.unload will also disconnect from a server, but not necessarily gracefully.

10.3 request

dde.request(item [, format [, timeout]])

dde.request gets data from the server.

If there is no conversation with a server in progress (started with [dde.start_conversation](#)), **dde.request** will return an error message.

item must be a valid item on the server, e.g. the address of a cell on an Excel sheet. It must be less than 255 characters in length (a DDE limitation).

format if present must be a valid format for data transfer from the application. See Section [11.1.1](#). If absent, plain text is assumed.

Unlike all other DDE transactions, described in the following sections, I can't figure out how to make a request transaction run asynchronously, so PowerPro will block while it's being processed by the server. If the server takes too long, the request will time out. The default time out period is 500 milliseconds. You can override that by specifying the **timeout** parameter, time in milliseconds.

See Section [11.2](#) "Values Returned by DDE-Related Services" for what the service returns.

10.4 execute

dde.execute(command)

dde.execute starts a transaction to execute a command on server

If there is no conversation with a server in progress (started with [dde.start_conversation](#)), **dde.execute** will return an error message.

command is a valid command that the server knows how to execute.

When you invoke **dde.execute**, a DDE execute transaction is started on the server, and control returns to your script. When the server finishes the transaction, it will inform the plugin. Call [dde.check_service_complete](#) to determine if the transaction has finished.

If there's a transaction still in progress from a previous service call (e.g. a poke), it will be abandoned, though I'd recommend ensuring any previous transaction has completed with a call to [dde.check_service_complete](#).

See Section [11.2](#) "Values Returned by DDE-Related Services" for what the service returns.

10.5 poke

dde.poke(item, data [, format])

dde.poke starts a transaction to poke data at server.

If there is no conversation with a server in progress (started with [dde.start_conversation](#)), **dde.poke** will return an error message.

item must be a valid item on the server, e.g. the address of a cell on an Excel sheet. It must be less than 255 characters in length (a DDE limitation).

data is the data you wish to poke at the **item**. It can be any length.

format if present must be a valid format for data transfer from the application. See Section [11.1.1](#). If absent, plain text is assumed.

When you invoke **dde.poke**, a DDE poke transaction is started on the server, and control returns to your script. When the server finishes the transaction, it will inform the plugin. Call [dde.check_service_complete](#) to determine if the transaction has finished.

If there's a transaction still in progress from a previous service call (e.g. an execute), it will be abandoned, though I'd recommend ensuring any previous transaction has completed with a call to [dde.check_service_complete](#).

See Section [11.2](#) "Values Returned by DDE-Related Services" for what the service returns.

10.6 advise

dde.advise(item, callbackCommand [, format])

dde.advise starts a transaction to start monitoring an item on the server. Normally the server will call

If there is no conversation with a server in progress (started with [dde.start_conversation](#)), **dde.advise** will return an error message.

If there's a transaction still in progress from a previous service call (e.g. an execute), it will be abandoned, though I'd recommend ensuring any previous transaction has completed with a call to [dde.check_service_complete](#).

item must be a valid item on the server, e.g. the address of a cell on an Excel sheet. It must be less than 255 characters in length (a DDE limitation).

format if present must be a valid format for data transfer from the application. See Section [11.1.1](#). If absent, plain text is assumed.

callbackCommand must be a valid PowerPro command. Typically it will be a call to a script, e.g. `.myScript@OnChange`. When you invoke **dde.advise**, a DDE advise transaction is started on the server, and control returns to your script. However, the server will inform the dde plugin whenever **item** requires (typically when it changes value). Every time the plugin is informed, it will invoke **callbackCommand**. See Section [11.1.2](#) for more details.

See Section [11.2](#) "Values Returned by DDE-Related Services" for what the service returns.

Once you start an advise transaction, be sure to stop it with [dde.advise_stop](#) (with the same **item** as you originally used to start it). [dde.stop_conversation](#) will also stop the advise, but not necessarily gracefully. **dde.unload** will do the same.

10.7 get_last_result

dde.get_last_result

dde.get_last_result returns the last result obtained from the server. Only meaningful to call it in the script invoked by [dde.advise](#) via its **callbackCommand**.

10.8 advise_stop

dde.advise_stop(item)

dde.advise_stop starts a transaction to stop monitoring an item on the server.

item must be a valid item on the server, e.g. the address of a cell on an Excel sheet.

If there is no conversation with a server in progress (started with [dde.start_conversation](#)), **dde.advise_stop** will return an error message.

See Section [11.2](#) "Values Returned by DDE-Related Services" for what the service returns.

10.9 check_service_complete

dde.check_service_complete() will return 1 if the last service call has completed, 0 if not, and the variable **dde_status** will be set to a status message ("OK" or something beginning "ERROR").

If the last service call was a [dde.request](#) or [dde.advise](#), **check_service_complete** will always return 1.

The **dde.check_service_complete()** service will never raise the PowerPro script error dialog, no matter whether you have `dde.error_dialog_on` in force or not.

Typically you would invoke the **dde.check_service_complete()** service from a timer, an event, or in a loop with `wait.for`, which would be started after you called a threaded service. So to start a threaded connection you might use the following script:

```
@ddeThreadedTestTimed
dde.connect_to_accnt("dde.server.com", "myusername", "mypassword", 25, 0, 0, 1)
Timer Set +c 0 0 2
timeoutCount = 0
Quit
```

I assume timer c's the "running" and "auto start" check boxes would be left unchecked.

Timer c would count down, and on reset would cause a script rather like the following to fire:

```
global sm_status
If (timeoutCount > 10) do
  Quit
Endif

if (dde.check_service_complete()) do
  if (not index(sm_status, "ERROR")) do
; send mail now
  endif
else
  timeoutCount = timeoutCount + 1
  Timer Set +c 0 0 2
endif

Quit
```

10.10 **error_dialog_on()**, **error_dialog_off()**

Some ini services can result in errors. For instance, you might try to read a non-existent ini file, or read a section/key pair that doesn't exist. In addition to such errors setting a PowerPro variable or returning a value with a status message the prefixed "ERROR:" (see previous section); they will also trigger the standard PowerPro script error dialog, allowing you to cancel all running scripts.

Under some conditions you might not want dde plugin errors to be treated as scripting errors, and you would therefore not want to see the PowerPro script error dialog. If that's what you want (maybe because you're testing for the presence or absence of a section/key pair), invoke **dde.error_dialog_off()**, or make sure the **raiseErrors** key in the configuration ini file is false or 0. Invoke **dde.error_dialog_on()** to turn error dialogs back on after you turn them off.

If you unload the plugin, its behaviour returns to the default in any subsequent call, i.e. errors on further dde service will cause the error dialog to pop up.

Invoking **dde.error_dialog_off()** only affects error dialogs appearing when a dde service call goes wrong. The normal Powerpro error dialog will appear if anything else goes wrong in Powerpro.

10.11 version

dde.version()

dde.version returns the plugin version number as four digits, an assumed decimal point before the last two digits.

10.12 help

dde.help()

dde.help opens the help file associated with the dde plugin, if

- the key [helpFileLocation](#) in the config file dde.ini/plugins.ini specifies a valid file path (rtf or chm or anything, really) location. If path isn't absolute, it's taken as relative to the PowerPro installation folder. Or, if the key [helpFileLocation](#) isn't present,
- the file dde_plugin_readme.chm, dde.chm, dde_plugin_readme.rtf or dde.rtf is found in either the PowerPro installation folder or the plugins subfolder thereof (the files are searched for in the order given).

10.13 config

dde.config(name_of_dde_file)

specifies a configuration ini file, with format, section and keys as described in section 9.1

The ini file can either be given as an absolute path, or a path relative to the folder returned by the pprofolder variable (generally the folder containing the currently running powerpro configuration file).

Returns "OK" if file and found and there are no keys with illegal values, or a message beginning "ERROR:" if there is one.

If you unload and reload a plugin, it's behaviour returns to the default or to that defined by a default config ini file (see section 9.1).

.

11.1 Common Service Parameters

11.1.1 format

format specifies the format in which the application will return or expects data. It is case insensitive and may be either a single letter or a word, as follows:

Message Priorities		
letter; or	word	means
t	text	plain text
c	csv	comma delimited
r	rtf	rtf

You may also specify any string that's the name of a valid clipboard format, providing of course the data produced or sent can be represented as a plain text string (as required by PowerPro).

11.1.2 callback

Most DDE transactions (except [dde.request](#)) eventually result in the server invoking a callback function. That function may in turn invoke a PowerPro callback command – typically a script call – provided as a callback argument to a DDE service call.

For the moment only the [advise](#) service uses a callback command.

A callback argument is a string which is a valid PowerPro command. It can be an invocation of a script or of and PowerPro command. It might look like this:

```
local successCmd = ".aScript@aLabel"  
dde.advise( "R2C2:R3C2", successCmd, "csv")
```

```
local successCmd = "*RunFile anotherScript(\"#\")"  
dde.advise( "R1C1", successCmd, "csv")
```

```
local sCmd = "." ++ scriptname ++ "@allOk"  
dde.advise( "R1C1", successCmd, "csv")
```

A callback command may include an *insertion marker*. This is hash (“#”) by default, but may be configured with the [insertionMarker](#) key of the dde.ini/plugins.ini configuration file. If there’s an insertion marker in a callback command, the service may insert data to return to PowerPro at that mark.

If the expected result is a string (as opposed to a number), you must provide the enclosing quotes for that string, so typically any use of the insertion marker # will show up in your callback argument as “\#”.

So for instance

```
local sCmd = "?%.dataArrived(\"#\")%  
dde.advise( "R1C1", sCmd, "csv")
```

will, everytime R1C1 changes, invoke the script file named dataArrived.powerpro or dataArrived.txt with arg(1) set to the changed data from the server.

However, that usually won’t work, because Excel seems to append a newline and carriage return to data it send back. So, since they’re not escaped, they’re real line breaks, meaning PowerPro sees a line termination before the final quote. Best in this case to use a callback without an insertion marker, and call [dde.get_last_result](#) in the callback script instead.

Be careful using callbacks to scripts that may be in the middle of execution. Remember that PowerPro isn’t itself multithreaded, so that if it happens to be executing aScriptfile.powerpro at the very instant a net service calls back to aScriptfile@aLabel, some strange stuff may happen. I have no idea what. You can probably minimise chances of collision by invoking whole script files, instead of a label within a script (which would increase chances that you will have invoked some other part of the same script file from within PowerPro at the same instance).

I've had problems with PowerPro crashes doing a dde.unload from a script invoked by a callback command. Avoid doing so, that's my advice.

11.2 Values Returned by DDE-Related Services

If a service succeeds, it will return "OK"; if it fails for some reason, it will return a string beginning "ERROR:"¹

Failed DDE API calls set an error code. I return the symbolic name of that code appended to the returned error message. The symbolic names are explained at this wonderful url:

<http://msdn.microsoft.com/library/en-us/winui/winui/windowsuserinterface/dataexchange/dynamicdataexchangemanagementlibrary/dynamicdataexchangemanagementreference/dynamicdataexchangemanagementfunctions/ddegetlasterror.asp?frame=true>

11.3 dde_ Variables

dde_status is set by [request](#) and [dde.get_last_result](#).

12.0 Restrictions

13.0 Possible Enhancements

At the moment you can have only one conversation with one server at a time, and only one transaction can be in progress at a time. In particular, although DDE architecture allows many [advise](#) transactions to be in progress at once, you can't do that in this version of the plugin. I can fairly easily extend the plugin to allow simultaneous connections to multiple servers and/or multiple simultaneous transactions

¹ The exception is [dde.request](#), which sets the variable **dde_status** to "OK" or "ERROR..." and returns the requested data; and [get_last_result](#), which only returns data.

14.0 Change History

.55: 18 November 2008

- fixed bug in start_conversation that permanently prevents further conversations if start_conversation fails.

.54: 13 November 2008

- un-enforced 255-character on executed data
- added version resource and a couple of icons
- added ddePluginFunctions.txt to be used as a file menu, perhaps merged with pprofuctions.txt.
- added ddePluginTestScriptEccoPro.powerpro (by Sheri Pierce) to sample scripts
- added ddePluginReadme.pdf

.53: 24 February 2006

- un-enforced 255-character on poked data

.52: 23 February 2006

- enforced 255-character string size limits on all arguments.

.50: 21 February 2006

- nada. First version.