

TPF Software News

Volume 2, Issue 2, Fall 2000

© 2000, TPF Software, Inc.

THIS ISSUE,
WE ASK

What If...?

What if your **debugging tool** was also a **design tool** that helped take the guesswork out of making enhancements to your TPF system? **Page 2.**

What if your **Assembler tool** was also a **C/C++ tool** that displayed linkmaps, variables, structures, and objects? **Page 3.**

Table of Contents

- 1 What If...?
- 2 **Bass** programmer asks *What If?* and uses TPF/GI to design enhancements
- 2 Trace Options Window saves address stops
- 3 New Linkmap Window makes Source View more powerful
- 4 New scripting feature allows programmers to automate TPF/GI
- 6 GI/FTP improves file download
- 6 New features a hit in new millennium
- 7 New Repeat Tool boosts productivity
- 8 Further product enhancements

What if your **test tool** was also **scriptable**? What if you could automate sending commands to the host, loading source files, running programs, and recording terminal input and output? **Pages 4-5.**

(In other words)

What if your **test tool** was **TPF/GI**?

Bass programmer asks *What If?* and uses TPF/GI to design enhancements

One day this year Jerri Peterson, of Bass Hotels and Resorts, realized something unexpected about TPF/GI.

TPF/GI is not just a program *testing* tool, she realized. It's a program *design* tool.

That Eureka moment led Jerri to coin a new term: *What If programming*—it's a concept she's eager to see adopted by other programmers who work with TPF/GI.

What If...?

Jerri's work includes designing new enhancements for the TPF system at Bass Hotels and Resorts. With the intricate way that TPF programs relate to each other, adding enhancements can be a daunting task.

TPF/GI is not just a program testing tool. It's a program design tool.

In fact, Jerri says, "Enhancing can be such an ordeal it's something you don't want to do. I just finished designing a project from hell. I would hate to try it without the GI."

One of the questions a designer must answer is *How does the system work now?* Once that is determined, the designer can decide where to hook in the necessary enhancements without breaking anything.

This is where Jerri finds TPF/GI's strength as a design tool: in its ability to explore the system, giving total control to the designer.

Using TPF/GI, Jerri says, "I can pinpoint exactly what programs need to be changed. I can stop in a program before it hits something, force it into some other code, change some indicators, jump out,

do something else.

"My estimates are so much more accurate because I know which programs need to change."

"I can see, 'Oh, this is what I need to do and this is how I need to do it.' For example, I can see that I won't have a record I need at this point in the processing. Or I might say, 'If only I had a core block here,'" Jerri says. "And with the GI you don't have to add any code, you can just create the core block."

The power that TPF/GI gives her to

explore is the origin of the phrase, *What If*. What if I need this record here, will I have it? What if I had a core block here? What if I make the code jump from A to B: will it work? TPF/GI gives programmers the power to find the answers to *What If* questions.

Accuracy

The ability that TPF/GI provides to find out answers and keep going allows Jerri to accomplish her design tasks faster—and more accurately.

"I can actually be more accurate for my designs for enhancement [using TPF/GI]," Jerri says. "My estimates are so much more accurate because I know which programs need to change."

"The GI really makes your estimates a lot more accurate and your designs more accurate," Jerri says. "Using the GI as a design tool you avoid 90% of the failure [you would experience otherwise]." ❖

Save address stops in Trace Options window

TPF/GI has found a way to make its Trace Options Window even easier to use: the Trace Options Window now remembers which address stops programmers have used in the past.

The Trace Options Window is TPF/GI's graphical method for setting up a trace—for example, to specify which programs to trace, which address stops to set, and which macros, instructions, and store locations to monitor.

In previous versions of TPF/GI, address stops were not saved across TPF/GI sessions.

But now, when programmers view the Trace Options, they are offered a list of address stops they have used previously. To the left of each address stop item is a checkbox. To reuse a prior address stop, programmers merely have to place a check in the appropriate checkbox. ❖

New Linkmap Window makes Source View more powerful

TPF/GI has added another powerful debugging aid to Source View: the Linkmap Window.

The Linkmap Window allows programmers to see the files and functions that make up a program, and to easily navigate to those files and functions.

Tree structure

The Linkmap Window presents information in a handy tree structure.

The first level of the tree contains the program name. The second level contains the files that the program is composed of. And the third level contains all the functions in each file, arranged alphabetically (Figure 1).

One strength of the Linkmap Window lies in its ability to let programmers add files to trace and navigate through their code.

To view a linkmap, programmers can select View>Linkmap... from the menubar and enter the name of a program in a dialog box. Or, even more conveniently, while stopped in Source View programmers can click a tool button, and the linkmap for the active program will be displayed.

Easy navigation

One strength of the Linkmap Window lies in its ability to let programmers add files to trace and navigate through their code.

Once a linkmap is displayed, programmers can add a file to Source View trace



Figure 1: The Linkmap Window uses a tree structure to display the files and functions that belong to a program.

by double clicking the file in the Linkmap Window. If the file has already been added to Source View trace, double clicking the file name will display the file.

Similarly, programmers can navigate to a function in their code by double clicking the name of the function in the Linkmap Window. The file in which the function resides will be downloaded and added to Source View trace if necessary, and then the file will be displayed with the beginning of the function highlighted.

Setting up trace on the fly: an example

A quick example will show how useful the Linkmap Window can be. Suppose a program is about to call a function named *myFunc()* that the programmer would like to trace into, but the programmer doesn't know which file contains the function. He or she can do a Find (Ctrl+F) on the Linkmap Window and search for the text *myFunc()*. When the correct item is found, a double click will add the control file to Source View trace, and the programmer can step into *myFunc()* to look for bugs.

PC ease of use

The Linkmap Window is another step by TPF/GI to bring the power of PC-based debugging tools to TPF programs. When displayed, the Linkmap Window docks beside the Local Variables window, making it easy for programmers to switch from one to the other by clicking folder-style tab at the tops of the windows. The Local Variables Window allows programmers to view and change local variables in their human readable formats. ❖

New scripting feature allows programmers to automate TPF/GI

In order to give TPF programmers an even more efficient way to test, TPF Software has introduced a new scripting feature into TPF/GI.

Scripts allow programmers to automate TPF/GI in dramatic ways. Here are some of the things programmers can do with scripts.

Set up testing. Programmers can use scripts to automatically set macro and instruction trace, add files to Source View trace, and run a program.

Run multiple message files. Message files are an earlier kind of scripting: they contain a series of input commands for terminals. Using the new scripting feature, programmers can run several message files in a row and automatically stop and start terminal logging and recording.

What is a script?

A script is a text file written in one of several scripting languages, most likely VBScript (Visual Basic script).

Scripts allow programmers to automate TPF/GI in dramatic ways.

The script contains one required function or subroutine, named *main*, although the programmer can code other functions as well. Scripts can also contain comments (see Example 1 on the next page).

The TPF/GI object model

Scripts do most of their work by calling on objects that TPF/GI makes available.

Don't let the word "object" sound threatening, however. Scripts are made up of

straightforward, procedural programming, as Example 1 on the next page demonstrates.

The available objects for scripting are listed in the side-bar on the next page. They include terminal objects such as ALC1, ALC2, LOC1, LOC2, PrimeCRAS, and GIConsole. There is also a Source View object (SView), a Run object, a CMS object, an App object, and a Windows object. More objects will be added in the future.

Using scripts

Scripts are controlled from the new Scripts Window (Figure 1), reached by selecting File>Scripts from the menubar.

To write a new script, programmers can click the New button on the Scripts Window. Once the programmer has selected a script name and script language, the Notepad applet will appear containing a main subroutine. Here the programmer can type in his code and save the script.

To run a script, the programmer simply double clicks the script name in the Scripts Window.

Scripts are powerful. Depending on the contents of the script, windows will appear and disappear, commands will be sent to the host, breakpoints will be set, and files will be loaded into Source View. As the script executes, the top of the Script Window provides a play by play description of each call to TPF/GI. ❖

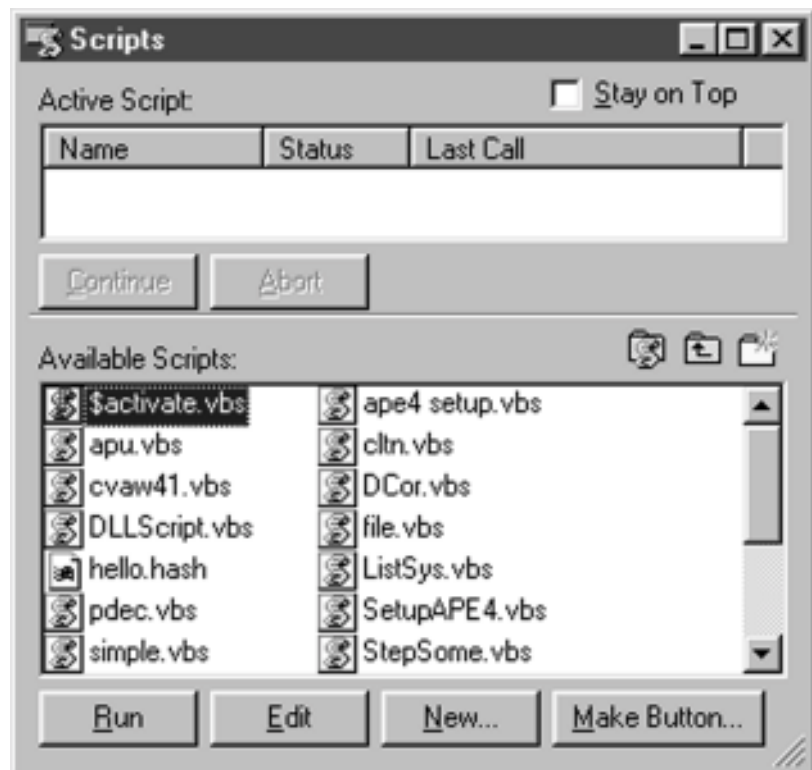


Figure 1: The new Scripts Window..

Script Commands

TPF/GI makes the following objects available to programmer scripts.

Terminal objects.

Scripts can call on objects named ALC1, ALC2, LOC1, LOC2, PrimeCRAS, and GIConsole to manipulate the TPF/GI terminals.

For example, to send input through ALC1, a programmer can write

```
ALC1.Input "command"
```

where *command* is some ALC command. To retrieve and use the output from a command, a programmer can write

```
S = ALC1.Input("command")
```

The variable *S* will now hold the output that the ALC terminal received as a result of the command.

Other commands that work with terminal objects are Show, Hide, Clear, Unlock, RunMsgFile, StartCapture, StopCapture, StartLog, and StopLog.

App Object

Scripts can call on the App object to find out about and change the current system, CPC, and ECB.

CMS Object

The CMS object allows scripts to execute CMS commands.

Windows Object

The Windows object allows scripts to refresh the contents of TPF/GI windows.

Source View Object

The Source View object (SView) allows scripts to add and remove source files, find out which source files are being traced, and set and remove Source View breakpoints.

Run Object

The Run object allows scripts to tell TPF/GI to step into, step through step out, run fast, run slow, and so forth. ❖

Scripting Examples

Example 1. A simple script that sets up a macro trace and runs a program.

```
Sub main
  ' Set up trace options with GI Console command
  GIConsole.Input "trace macro all"

  ' Run a program using PrimeCRAS
  '(could have used ALC1, ALC2, LOC1, LOC2)
  PrimeCRAS.Input "ZDSYS"

  ' We will now be stopped at the first macro
End Sub
```

Example 2. A script that demonstrates setting up a Source View trace.

```
Sub main
  ' $getusr is an exec that links to
  ' Frank's disk as the J disk
  GIConsole.Input "$getusr frank"

  ' loaddlm is an exec that loads a program
  GIConsole.Input "loaddlm APE4C0"

  ' Add several Source View control files
  SView.AddCtlFile "ape4a0 cmstpfct j"
  SView.AddCtlFile "cape4a0 cmstpfct j"
  SView.AddCtlFile "dape4a0 cmstpfct j"

  ' Force an ECB into program
  GIConsole.Input "create nomsg APE4"

  ' We now will be stopped in the first control file
End Sub
```

Example 3. A script that demonstrates running multiple message files.

```
Sub main
  ALC1.Show

  ' Start log and run first message file
  ALC1.StartLog "c:\my documents\alcl-a.log"
  ALC1.RunMsgFile "HOST:zmsgs input a"
  ALC1.StopLog

  ' Run second message file, logging to a second log file
  ALC1.StartLog "c:\my documents\alcl-b.log"
  ALC1.RunMsgFile "HOST:kmsgs input j"
  ALC1.StopLog

  ' Run third message file. This input file is on the PC!
  ALC1.StartLog "c:\my documents\alcl-c.log"
  ALC1.RunMsgFile "PC:c:\jmsgs.input"
  ALC1.StopLog

  ALC1.Hide
End Sub
```

Continued on p. 8

GI/FTP improves file downloads

TPF Software has added a new file transfer facility to TPF/GI that uses the FTP protocol.

The new facility, named GI/FTP, will allow users to transfer files to and from the host more easily, and will pave the way for integrating TPF Software products with non-CMS file systems.

Currently, GI/FTP has been incorporated into TPF/GI when the user selects File>Download to PC... from the TPF/GI menubar.

Latest look and feel

As you can see from Figure 1, the GI/FTP download dialog box provides the look and feel of the latest Microsoft products.

An "Outlook-style" bar on the left side of the dialog allows users to select the source of the files they will browse. A drop-down box at the top of the dialog allows users to select a PC folder or a VM disk depending on where they are browsing.

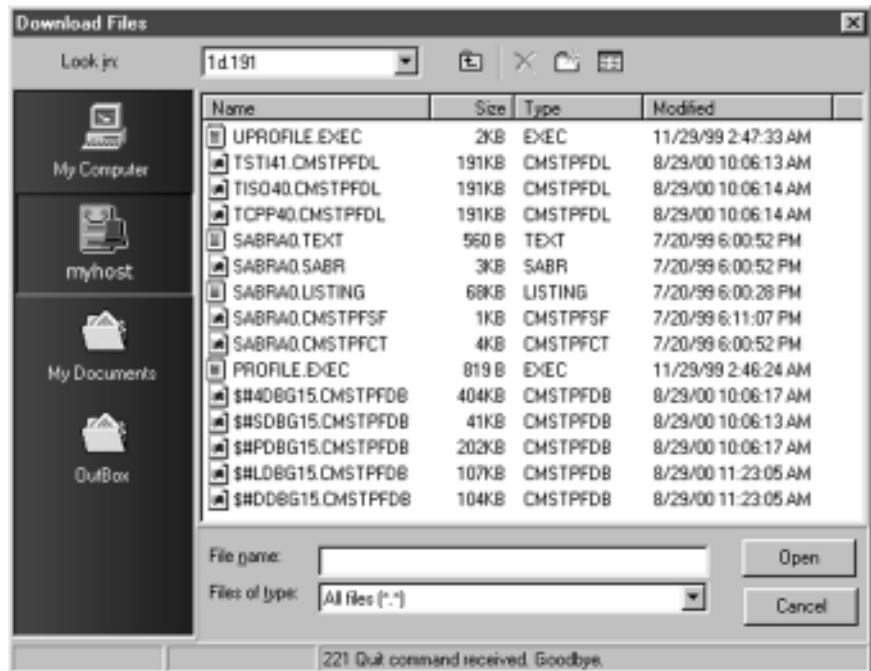


Figure 1: The GI/FTP download dialog box.

Users can elect to view files on the host or on their local PCs. If the user selects a host file, it is downloaded to the PC. ❖

New features a hit in new millennium

TPF/GI's latest features have programmers across the country singing its praises.

Jerri Peterson of Bass Hotels and Resorts praises the Program Flow feature, which was new with the current version of TPF/GI. The Program Flow feature uses a tree of icons to help programmers visualize the flow of nested calls from one program to another within their applications.

"That program flow tree is just a little marvel," Jerri says. "I love it. It shaved an hour off my work. It's a great asset for a design tool."

C++ support

Dan Maxfield, also of Bass Hotels, has been working with TPF/GI's support for debugging and testing C++ code.

"TPF/GI stands alone as the only TPF product that provides a modern testing environment for C++," Dan says. "Its Source View traces jump effortlessly from Assembler, to C, to C++, and the Local Variables/DFDisplay functions really deliver the goods."

"TPF/GI stands alone"

Brian Lowderman, of Worldspan, has also been working with TPF/GI's C++ support, and he agrees. The new Local Variable features, he says, "will make the job easier." He cites the fact that programmers can use TPF/GI "like the PC editors. That's what everybody's familiar with if they've done any coding in college."

In particular, Brian is impressed with the way the variable values update in the Local Variables window. "It is nice to see that when I'm switching between functions it does the refresh and finds the variables," he says. "The color change is definitely helpful."

TPFAR support

Wade Brabits of Worldspan has been using TPF/GI's new support for TPFAR.

"We don't need to test the TPFAR process itself," Wade says. "We call utilities that use TPFAR and we need to test all that stuff that comes after the TPFAR process. The utilities are written in C."

"Before, it wasn't possible to test any code that got hit after a TPFAR process was called," Wade says. "[TPF/GI] is the only TPF-friendly test tool for C programs." ❖

New Repeat Tool boosts productivity

The new Repeat Tool is a window that remembers certain actions a programmer takes in TPF/GI. It then allows the programmer to repeat any single action with a simple double click.

The Repeat Tool not only aids the productivity of programmers, it also demonstrates the power of TPF/GI's plug-in tools Application Programming Interface (API), since the Repeat Tool was written as a plug-in tool.

Using the Repeat Tool

Programmers first need to install the Repeat Tool (select Help>What's New from the TPF/GI menubar to learn how to do this). They can then select Tools>Repeat from the menubar to display the Repeat Tool window.

To repeat any action, the programmer simply double clicks it.

The Repeat Tool remembers the following actions:

- GI Console messages
- Adding of a Source View control file
- Setting of a Source View breakpoint
- Removal of a Source View breakpoint
- Loading a program

Every time one of these actions is performed, the action appears at the top of the Recent page of the Repeat Tool window (Figure 1).

To repeat any action, the programmer simply double clicks it (or selects it and presses Enter). To remove the action, the programmer presses the Delete key.

Organizing and Saving

In order to organize and permanently save

actions, programmers can transfer them to folders on the Favorites page of the Repeat Tool window.

To save the action to Favorites, the programmer can right click the action and select Move to Folder from the right click menu.

Works with Scripting

It's true that TPF/GI has added scripting to let programmers automate their work with TPF/GI, but some frequently repeated actions may not seem important enough to script. The Repeat Tool lets even these less important actions be automated to some degree.

And the Repeat Tool does work with scripting. Once a programmer has transferred a set of actions to a folder on the Favorites page, he or she can create a script by simply right clicking the folder and selecting Export to Script... from the right click menu. A VBScript file is produced which can be executed from TPF/GI's Script Window. ❖

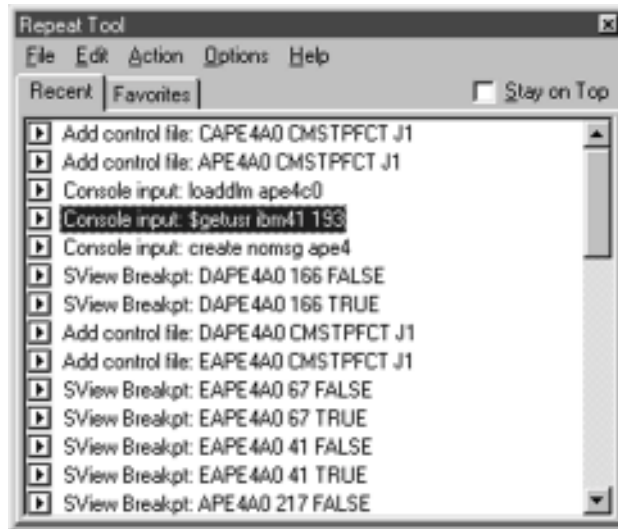


Figure 1: The Recent page of the Repeat Tool window..

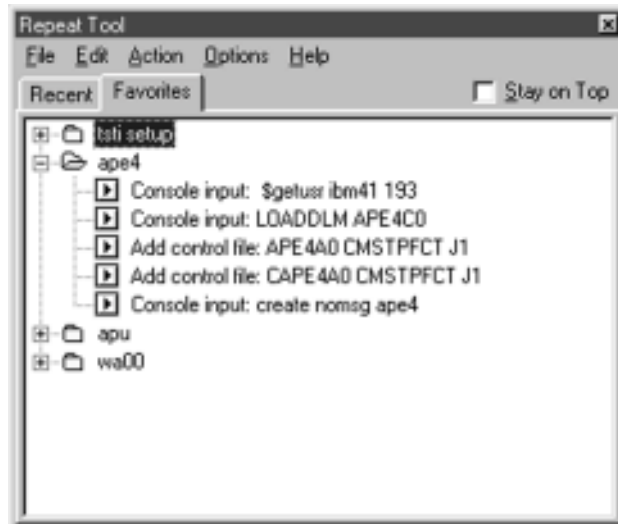


Figure 2: The Favorites page of the Repeat Tool window..

Further product enhancements

Below is a partial list of other enhancements made to TPF Software products.

Compiler Support

CMSTPF and TPF/GI now support versions 2.4 and 2.6 of the C and C++ OS/390 Compiler. Programs compiled under these versions can be traced with Source View. Local variables for programs compiled under these versions can be viewed with TPF/GI.

DLL and Main() Support

CMSTPF and TPF/GI now support calls to DLLs. Programmers can now trace calls to DLLs that import and export functions. Additionally, DLMs that are defined with Main() support can now be loaded and tested under CMSTPF.

Put 12 Support

CMSTPF and TPF/GI now support Program Update Tape 12 (Put 12).

TCP/IP Support

CTFS is being updated to include support for TCP/IP. With the TCP/IP support, TPF programs that use native TCP/IP sockets can now be tested under CMSTPF.

When a TCP/IP socket request is made, CMSTPF will intercept the socket request and send the request across to TPF (VPARS) for processing. Additionally, this same support is also being added to TTFS.

Line-column numbers in ALC log

The ability to include line and column numbers in ALC logs in TPF/GI has been restored. The line and column numbers are especially useful to programmers who need to communicate the position of an error to another programmer.

One-click Assembler Expressions

Programmers can now view Assembler expressions and DSects without passing through the Edit Expression dialog box. The Assembler Expression Facility allows programmers to watch and edit the results of assembler expressions while they step through their assembly code in TPF/GI's Source View.

Retrieve list improvements

The retrieve lists in the ALC, GI Console, and Prime CRAS windows can now be cleared, and programmers can set the maximum number of items that each retrieve list holds.

now turns off machine instruction trace for all Source View control files. This change brings machine instruction trace behavior in line with user expectations.

Easy tool button removal

TPF/GI allows users to customize toolbars by placing additional buttons on them. Now, removal of tool buttons has been made easier. Users simply right click a tool button and select "Remove Item from toolbar" from the right click menu; the button is removed.

Tool button separators

Users can now place separator items on toolbars to organize their tool buttons. ❖

CMSTPF & TPF/GI now support calls to DLLs.

Close Machine window, stop trace

Closing the Machine Instructions window

Script examples continued

Example 4. A script that performs different actions based on the active system.

```
Sub main

  If App.ActiveSystemID = "G" Then

    ' Take actions specific to system G
    GIConsole.Input "System G specific command"

  Else

    ' Take actions specific to other systems
    MsgBox "This script works only with system G"

  End If

End Sub❖
```



Contact TPF Software

Web Site www.tpf-software.com

Email info@tpf-software.com

Telephone 919-676-5501

Address TPF Software, Inc.
8729 Gleneagles Drive
Raleigh, NC USA 27613-5419

TPF Software News Editor: Ed Jordan