PAMS
TCP/IP Queue Adapter

Novmber 1990

This document describes the features, restrictions, and use of the PAMS TCP/IP Queue Adapter for providing PAMS functionality on TCP/IP connected systems.

Revision/Update Information:	This is a new document.
Operating System and Version:	VAX/VMS V5.2 or higher ULTRIX V3.0 or higher
Software Version:	VAX PAMS V2.5 VMS/Ultrix Connection V1.2 or higher

October 1990

The information is this document is subject to change without notice and should not be construed as a commitment by Digital Equipment Corporation. Digital Equipment Corporation assumes no responsibility for any errors that may appear in this document.

The software described in this document is furnished under a license and may be used or copied only in accordance with the terms of such license.

No responsibility is assumed for the use or reliability of software on equipment that is not supplied by Digital Equipment Corporation or its affiliated companies.

The following are trademarks of DIGITAL Equipment Corporation:

VAX
VAXcluster
VMS/ULTRIX Connection
VMS
DECnet
ULTRIX
DEC
digital™

Ethernet is a registered trademark of Xerox Corporation.

UNIX is a registered trademark of AT&T in the USA and other countries.

This document was prepared using VAX DOCUMENT, Version 1.2

Contents

EXAMPLES

FIGURES

TABLES

1 Introduction

This document describes the *PAMS TCP/IP Queue Adapter* software, (also referred to in this document as the *TCP/IP Adapter*), a software package included in the *VMS-PAMS* product. The TCP/IP Adapter will allow most *PAMS* client applications, with minimal changes, to use TCP/IP communications to access the message bus through a host adapter process. The application client can reside on the same host system as the TCP/IP adapter, or on a remote system connected by a TCP/IP datalink to the host. Multiple TCP/IP clients can communicate with each other, or with other remote systems, by using multiple TCP/IP adapters on the host system. The client system(s) must support TCP/IP socket calls and the C programming language. Digital's *VMS/Ultrix Connection* product is required on all VAX/VMS systems that are used as message bus host systems running TCP/IP Adapters.

This document will describe the architecture, features, and limitations of using the TCP/IP Adapter, as well as provide instructions for creating or modifying client applications to use it.

2 TCP/IP Queue Adapter Overview

2.1 Description

The TCP/IP Queue Adapter allows a client to access a *PAMS* message bus through a TCP/IP communications link with a host process. Client applications may communicate with each other, or with multiple remote systems, by using multiple TCP/IP Adapters. Although the client application may reside on the same host system as the TCP/IP Adapter, the more common application is for the client to run on a remote system that is not a member of the message bus, but can communicate to the host system through a TCP/IP datalink.

The TCP/IP Queue Adapter consists of two main functional components: the host *TCP/ IP Adapter Process*, and the client *TCP/ IP Adapter API* (Application Programming Interface) functions. Instead of *PAMS* service functions, the client program is built with the TCP/IP Adapter API. The TCP/IP API functions have the same entry parameters as their corresponding *PAMS* functions, but they forward the parameters to the TCP/IP Adapter running on a host system. The TCP/IP Adapter changes this information into the *PAMS* command, executes it, and then returns the results to the client's TCP/IP API. The response is formatted into valid return information and is then passed back to the calling client application. To the client, it appears as if the called function had execute on the client's local system. Figure 2–1 depicts this logic flow.

2.2 Functionality

The TCP/IP Adapter provides message bus access to systems that support a TCP/IP connection to a host *VMS-PAMS* system. Existing *PAMS* client applications should be able to use the TCP/IP API with little or no changes.

The TCP/IP Adapter and API provides support for the following *PAMS* functions:

- **PAMS_DCL_PROCESS**
- **PAMS_PUT_MSG**
- **PAMS_GET_MSGW**
- **PAMS_SET_SELECT**
- **PAMS_CONFIRM_MSG**
- **PAMS_EXIT**

Running multiple TCP/IP Adapter Processes on the host *PAMS* system will allow tasks on different remote TCP/IP systems to communicate with each other through the message bus, as shown in Figure 2–2.

Figure 2–1 Logic flow of a TCP/IP Adapter Client

```
          .-  .---------------------------.
          |   |      User Application     |
          |   +--                       --+
          |   |     PAMS function call    |
          |   +---------------------------+  -.
 Client --+   |   Format PAMS call into   |   |
          |   |      a TCP/IP message     |   |
          |   +--                       --+   +-- TCP/IP API
          |   |  Send TCP/IP message to   |   |   Functions
          |   | TCP/IP Adapter and await  |   |
          |   |         Response          |   |
          '-  '---------------------------'  -'
                            \
                            /
                            \
                            /
                            \
                            V
          .-  .---------------------------.
          |   |  Receive TCP/IP message   |
          |   +--                       --+
          |   |  Format message into the  |
          |   |    Requested PAMS call    |
          |   +---------------------------+  -.
TCP/IP  --|   |     Perform PAMS call     |   +-- PAMS
Adapter   |   +---------------------------+  -'
          |   |   Format results into a   |
          |   |       TCP/IP message      |
          |   +--                       --+
          |   |   Send TCP/IP message to  |
          |   |        Client's API       |
          '-  '---------------------------'
                            \
                            /
                            \
                            /
                            \
                            V
          .-  .---------------------------.  -.
          |   |  Receive TCP/IP Response  |   |
          |   +--                       --+   |
          |   | Format message into PAMS  |   |
 Client --+   |       Return values       |   +-- TCP/IP API
          |   +--                       --+   |   Functions
          |   |   Return to application   |   |
          |   +---------------------------+  -'
          |   |   Application program     |
          '-  '---------------------------'
```

2.3 Requirements

Currently, the TCP/IP Adapter Process only runs under the VAX/VMS operating system. DECnet is not required for the TCP/IP Adapter, even if multiple adapters are being used to "connect" multiple remote TCP/IP systems, but *PAMS* may require DECnet if there are messages to groups on other nodes not connected by TCP/IP links.

The TCP/IP Adapter Process requires the following:

- *VAX/ VMS* V5.2 or higher
- *VMS/ Ultrix Connection* V1.2 or higher

Figure 2–2 Using Multiple TCP/IP Adapter Processes

```
                                                             .
                                                            / \
.--------------------.  .------------------------------.   /   \
|       Ultrix       |  |                              |  /_   _\
| .---------------.  |  |   .------------------------. |    |M|
| | TCP/IP Client |<------->| TCP/IP Adapter Process |<---->|E|
| '---------------'  |  |   |       Port 1050        | |    |S|
|     System #1      |  |   '------------------------' |    |S|
'--------------------'  |                              |    |A|
                        |   Host VAX/VMS PAMS System   |    |G|
.--------------------.  |                              |    |E|
|        Unix        |  |                              |    | |
| .---------------.  |  |   .------------------------. |    |B|
| | TCP/IP Client |<------->| TCP/IP Adapter Process |<---->|U|
| '---------------'  |  |   |       Port 1055        | |    |S|
|     System #2      |  |   '------------------------' |  .-' '-.
'--------------------'  '------------------------------'  \     /
                                                           \   /
                                                            \ /
                                                             '
```

- *VMS-PAMS* V2.5 or higher.

For building client applications, the TCP/IP API requires:

- A **C** language compiler. If using *VAX C*, this must be V3.0 or higher.
- TCP/IP socket communication functions.
- If using *Ultrix*, it should be version 3.0 or higher. *RISC Ultrix*) should be version 4.0 or higher. Other versions or operating systems may require source file modifications to the TCP/IP API.

TCP/IP Client applications do not require *PAMS* or *DECnet* to use the TCP/IP API functions to communicate with the TCP/IP Adapter.

It should be possible to use the TCP/IP API and the user client application under any operating system that supports TCP/IP socket communications and a **C** language compiler. Effort has been made to make the source as portable as possible. There may be cases, however, where special editing of the user client or the TCP/IP API source files may be necessary. This will depend on the operating system and the compiler used.

2.4 Components

The TCP/IP Adapter package consists of the following logical components:

1 The TCP/IP Adapter Process for the *PAMS* VAX/VMS host system. This includes:

   - The **PAMS_TCPIP.OLB** object library (containing the binary image of the TCP/IP Adapter Process).
   - The TCP/IP Adapter Process build procedure.
   - The TCP/IP Adapter Process startup procedure.

2 The TCP/IP API for client applications written under VMS and non-VMS systems. These files include:

- The TCP/IP Adapter communication functions source files.
- The TCP/IP Adapter *PAMS* entry functions source files.
- The include files and **PAMS_TCPIP.TLB** text library.
- The link options file for VMS client applications.

3 Documentation files and sample client applications.

The following is a summary list of all files that are part of the TCP/IP Adapter:

- **PAMS$LIB:**
  - **PAMS_TCPIP.OLB** - Contains the binary image of the TCP/IP Adapter Process, and object files of the TCP/IP API for building TCP/IP clients under VAX/VMS.
  - **PAMS_TCPIP.TLB** - Contains the TCP/IP Adapter include files for building clients applications under VAX/VMS.
  - **PAMS_TCPIP.OPT** - Link options file for building a TCP/IP client under VAX/VMS. Use this instead of the **PAMS.OPT** file.
- **PAMS$DOC:*TCP*.*** - *PAMS* TCP/IP Adapter documentation files.
- **PAMS$EXE:PAMS_TCPIP_ADAPTER.EXE** - The TCP/IP Adapter process executable.
- **PAMS$EXAMPLES:TCPIP_ADAPTER.DIR** - Directory containing sample TCP/IP Adapter client and test programs applications:
  - **samp_tcp.c** - Sample TCP/IP Client program derived from the *VMS-PAMS* **SIMPLE_CLIENT.C** example.
  - **tcptest.c** - Another TCP/IP client program, but also includes some code for testing TCP/IP Adapter communications.
- **PAMS$TCPIP_ADAPTER:**
  - ***.h** - The include files for building TCP/IP Client applications on non-VMS systems.
  - **p_tcpfnc.c** - Source file for the TCP/IP API communications functions; for building TCP/IP Client applications on non-VMS systems.
  - **p_tcpip.c** - Source file for the TCP/IP API *PAMS* entry functions; for building TCP/IP Client applications on non-VMS systems.
  - **PAMS$TCPIP_ADP_STARTUP.COM** - Command procedure for starting the TCP/IP Adapter process.
  - **PAMS$BUILD_TCPIP_ADP.COM** - Command procedure for building the TCP/IP Adapter executable image.

- **PAMS$INSTALL_TCPIP_ADP.COM** - Command procedure to copy separate kit directories into *PAMS* directories. This procedure is not normally needed.

These files are listed and described in greater detail in the next chapters.

2.5 Installation

The TCP/IP Adapter comes as part of the *VMS-PAMS* distribution and gets installed during the product installation procedures. After installation, the following items should be verified:

1. That the logical **PAMS$TCPIP_ADAPTER:** is defined and points to the location containing the command procedures and the include files for the TCP/IP Adapter API.
2. That **PAMS$TCPIP_ADP_STARTUP.COM** has been copied to **SYS$MANAGER**.
3. That all the files and directories listed in Section 2.4 are correct.
4. Build the **tcptest** program, as described in the *TCP/ IP Adapter Client Applications* chapter of this document.
5. Start the TCP/IP Adapter Process, as described in *TCP/ IP Adapter process* chapter.
6. Run the **tcptest** program, as described in the *TCP/ IP Adapter Client Applications* chapter and compare the results.

If it becomes necessary to rebuild the TCP/IP Adapter Process (**PAMS_TCPIP_ADAPTER.EXE**), this can be done by executing the **PAMS$TCPIP_BUILD_ADP.COM** as described in the next chapter.

Copying and installing the TCP/IP Adapter API on other systems is described in *TCP/ IP Adapter API* chapter.

3 The TCP/IP Adapter Process (PAMS_TCPIP_ADAPTER.EXE)

3.1 Overview

The TCP/IP Adapter normally runs as a VMS background process and waits for a TCP/IP connection request from the client. The connection request occurs automatically when the client calls the **PAMS_DCL_PROCESS** function in the TCP/IP Adapter API. The TCP/IP Adapter completes the connection, calls **PAMS_DCL_PROCESS** as specified by the client application, and returns the results back to the client. TCP/IP messages are sent from the client to the TCP/IP adapter and back as needed. The connection is terminated when the client application or TCP/IP Adapter terminates, or the client calls the **PAMS_EXIT** function in the API. The TCP/IP Adapter then waits for a new connection request.

Each TCP/IP Adapter can serve any TCP/IP client, but can only handle one connected client at a time. A client attempting to connect to a TCP/IP Adapter process that is already in use may appear to "hang" until that TCP/IP Adapter process is available. Once free, the connection request will be processed and the client application will "wake up".

The TCP/IP adapter is designed to tolerate or recover from most errors. In the event that an unexpected and unrecoverable error situation occurs that terminates the TCP/IP Adapter Process, an operator notification message will be sent to the system's console terminal (OPA0:) and (optionally) mailed to a specified user account. The error condition itself, or any output produced by setting the **PAMS$DEBUG** logical, can be (optionally) captured in a log file.

3.2 Building the TCP/IP Adapter Process

The **PAMS$BUILD_TCPIP_ADP.COM** procedure in the **PAMS$TCPIP_ADAPTER** directory will (re)build the TCP/IP Adapter executable image **PAMS_TCPIP_ADAPTER.EXE**. The *PAMS* installation procedures will automatically execute this procedure during product installation, but it can be re-run manually at anytime. Customizing *PAMS* or installing patch kits may require a rebuild of the TCP/IP Adapter. Example 3–1 demonstrates this procedure.

Example 3–1 Sample rebuild of the PAMS_TCPIP_ADAPTER.EXE image

```
$ @PAMS$TCPIP_ADAPTER:PAMS$BUILD_TCPIP_ADP

Building TCP/IP Queue Adapter image...
%COPY-S-COPIED, PAMS_TCPIP_ADAPTER.EXE;1 copied to DISK$PMB:[PA
MSV30.EXE]PAMS_TCPIP_ADAPTER.EXE;3 938 blocks)
TCP/IP Queue Adapter build completed

$
```

3.3 Starting the TCP/IP Adapter Process

Before the TCP/IP Adapter process can be started, the *VMS/ Ultrix Connection UCX* software must be installed and started. Complete details for are provided in the *VMS/ Ultrix Connection* documentation. For the TCP/IP Adapter, it is only necessary to start the INET processes.

Once the INET is started, a TCP/IP port number must be selected. Port numbers range from 1 to 65535. Any port number can be used, but some ports are reserved for specific tasks (ports 1-255), some require privileges (ports 1-1023), and some may already be in use. In addition to selecting a port number, the local system must be defined in the *UCX* database. Consult the *VMS/ Ultrix Connection* System Management documentation for complete details.

Once a port has been selected, starting the TCP/IP Adapter process is just a matter of executing the **PAMS$TCPIP_ADP_STARTUP.COM** procedure and supplying the port number:

Example 3–2 Starting the TCP/IP Adapter Process

```
$  @SYS$MANAGER:PAMS$TCPIP_ADP_STARTUP port

Starting TCP/IP Queue Adapter in group nn, on port port
%RUN-S-PROC_ID, identification of created process is nnnnnnnn
```

@SYS$MANAGER:PAMS$TCPIP_ADP_STARTUP port [mail-dest] [output] [pams-log]

port

TCP/IP port number to use for socket connection.

mail-dest (optional)

Username to receive a mail message notification if the TCP/IP Adapter process terminates abnormally.

output (optional)

Output file to use for SYS$OUTPUT; default is NL:. If specified, the output file will contain any error messages that occurred if the TCP/IP Adapter process terminates abnormally. Also, any output produced by setting the PAMS$DEBUG logical will also appear here.

pams-log (optional)

Sets the **PAMS$DEBUG** logical to the specified value before starting the TCP/IP Adapter. If the *output* parameter was specified, all output produced will be captured in the log file. Consult the *PAMS* documentation for further information concerning the **PAMS$DEBUG** logical and possible values.

Note: When specifying the third parameter (*output*) without the second (*mail-dest*) parameter; a double-quote ("") must be used in place

of the second parameter. (Ex: @SYS$MANAGER:PAMS$TCPIP_ADP_STARTUP 1025 " " ADAPTER.LOG). This same rule applies when specifying the fourth parameter without using the second or third parameter.

Once started, the program runs as a background process with a process name of **TCPADP_ggg-pppp**, where *ggg* is the group number and *pppp* is the port number.

3.3.1 Interactive Execution

When debugging clients applications, it is sometimes easier to examine the output of the TCP/IP Adapter (produced by setting **PAMS$DEBUG** or encountering a non-recoverable error) by running it as an interactive terminal session rather than as a background process.

1. Log into a privileged account on the desired VAX/VMS node and enable privileges. (Note: *VMS/ Ultrix Connection* INET must be running)).
2. Set your *PAMS* group and make sure that the **PAMS$*** logicals are set as needed.
3. Set the logical **PAMS$DEBUG** if desired.
4. Define a global symbol for the adapter as follows:

   $ TCPIP_ADAPTER == "$PAMS$EXE:PAMS_TCPIP_ADAPTER"
5. Start the TCP/IP Adapter for the desired port by typing the symbol name, followed by the desired port number (as shown in example Example 3–3).

Example 3–3 Starting the TCP/IP Adapter Process Interactively

```
$  SET PROCESS/PRIV=ALL
$  TCPIP_ADAPTER == "$PAMS$EXE:PAMS_TCPIP_ADAPTER"
$  DEFINE PAMS$DEBUG "TRACE"
$  TCPIP_ADAPTER port

Message Bus TCP/IP Adapter Vn.n-nn, protocol revision nn

Port number specified = port
Host address of NODE is n.n.n.n

   .
   .
   .
```

A TCP/IP Adapter running interactively can be aborted by using *Control/ Y* and typing *EXIT* at DCL. If the image appears to be "hung" after typing *EXIT*, repeat the sequence again. Because of the background communication connections that may be in use; the user may have to perform the "*Control/ Y* and *EXIT*" sequence twice to finally terminate the image.

4 TCP/IP Adapter API

4.1 Description

The TCP/IP Adapter API is used in place of the message bus functions and must be installed on the client's computer system. The TCP/IP API requires TCP/IP communication socket functions and a **C** language compiler.

4.2 TCP/IP Socket Communication files

You will need to locate the functions and include files on the client's computer system that support the TCP/IP socket communications. The include files needed are shown in Table 4–1. These files contain different structures and definitions per operating system; **DO NOT COPY OR USE THE TCP/IP SOCKET INCLUDE FILES OF ONE OPERATING SYSTEM ON ANOTHER**.

Table 4–1 TCP/IP Socket communication include files needed

File	Ultrix/Unix Filename	VMS Filename (VMS/Ultrix Connection)
types.h	sys/types.h	SYS$LIBRARY:TYPES.H
socket.h	sys/socket.h	SYS$LIBRARY:SOCKET.H
in.h	netinet/in.h	SYS$LIBRARY:IN.H
netdb.h	netdb.h	SYS$LIBRARY:NETDB.H
inet.h	arpa/inet.h	SYS$LIRBARY:INET.H
(uio def)	sys/uio.h	SYS$LIBRARY:UCX$INETDEF.H

If the client is to be run under VMS, the *VMS/ Ultrix Connection* product will be required.

Warning: If you are using *VAX C* version 3.0 or higher and *VMS/Ultrix Connection* version 1.2 together, a structure change to one of the include files may be required. Consult the *VMS/Ultrix Connection Release Notes* for details. The TCP/IP Adapter API will not function correctly without this modification if it is required.

4.3 TCP/IP Adapter API files

The TCP/IP Adapter API consists of the following files for non-VMS systems that need to be copied to the client's system:

- **PAMS$TCPIP_ADAPTER:*.h** - This includes:
  - **p_tcpip1.h**
  - **p_tcpip2.h**
  - **p_entry.h**
  - **p_process.h**
  - **p_group.h**

  - **p_typecl.h**
  - **p_return.h**
  - **p_aretur.h**
  - **p_symbol.h**
  - **sbs_msg_def.h**
  - **availmsg.h**
- **PAMS$TCPIP_ADAPTER:*.C** - This includes:
  - **p_tcpfnc.c** - TCP/IP Adapter Communications API.
  - **p_tcpip.c** - TCP/IP Adapter *PAMS* enpty point API
- **samp_tcp.c** and **tcptest.c** - Sample client and TCP/IP Adapter API test program (located in subdirectory of **PAMS$EXAMPLES**).

Clients on VMS only need the following files in **PAMS$LIB:**:

- **PAMS_TCPIP.TLB** - TCP/IP Adapter include files.
- **PAMS.TLB** - *PAMS* include files.
- **PAMS_TCPIP.OLB** - The TCP/IP Adapter API routines.
- **PAMS_TCPIP.OPT** - Link options file

4.4 Compiling the TCP/IP API functions

The source files **p_tcpfnc.c** and **p_tcpip.c** need to be compiled into object files that later can be linked into the client application. These files have been coded for portability and should compile without modification on *VMS*, *Ultrix*, and most other operating systems. If modifications are required, changes should be kept to a minimum and be as simple and straight-forward as possible. Changes should be made with "backward compatibility" in mind, possibly by using *#ifdef* and *#ifndef* statements around code changes.

Note: **Modifications must be restricted to the p_tcpip.c, p_tcpfnc.c, p_tcpip1.h, and tcpip2.h files. No changes should be made to any other TCP/IP Adapter include file (*.h), as it will inhibit code portability and may inhibit or alter *PAMS* functionality. Modified versions of p_tcpip1.h and p_tcpip2.h should also be replaced in the VMS PAMS$LIB:PAMS_TCPIP.TLB text library as modules PAMS_C_TCPIP1 and PAMS_C_TCPIP2.**

Under VMS, it should only be necessary to use or copy the four PAMS$LIB: files listed in the previous section. If circumstance demands that the TCP/IP Adapter API functions do require re-compiling from the source files in **PAMS$TCPIP_ADAPTER:**, Example 4–1 shows how to recompile and replace the library functions. Note that if the TCP/IP Adapter API functions are rebuilt, it may be necessary to also rebuild the TCP/IP Adapter Process.

Warning: **The PAMS_TCPIP.OLB library file also contains the binary image of the TCP/IP Adapter Process. Do NOT rebuild the library by**

deleting the original without first extracting the TCP/IP Adapter Process module.

Example 4–1 Rebuilding the VMS TCP/IP API library modules

```
$ SET DEF PAMS$EXAMPLES
$ SET DEF [.TCPIP_ADAPTER]
$ LIBR/EXTRACT=PAMS_TCPIP_ADAPTER/OUT=PAMS_TCPIP_ADAPTER -
 PAMS$LIB:PAMS_TCPIP
$ CC P_TCPIP+PAMS$LIB:PAMS_TCPIP.TLB/LIB+PAMS$LIB:PAMS.TLB/LIB
$ CC P_TCPFNC+PAMS$LIB:PAMS_TCPIP.TLB/LIB+PAMS$LIB:PAMS.TLB/LIB
$ LIBR/REPL PAMS$LIB:PAMS_TCPIP P_TCPIP
$ LIBR/REPL PAMS$LIB:PAMS_TCPIP P_TCPFNC
$ LIBR/REPL PAMS$LIB:PAMS_TCPIP PAMS_TCPIP_ADAPTER
```

5 TCP/IP Adapter Client Applications

5.1 Overview

All of the structures and definitions in the *VMS-PAMS* product (or other *PAMS* products) are available in the TCP/IP Adapter API include files. These files have different filenames and internal logic however, due to the need for functionality on non-VMS systems and compatibility with *PAMS* products for non-VMS systems (such as *PC-PAMS*). Table 5–1 provides a cross-reference between the files and their *VMS-PAMS* counterparts.

Table 5–1 TCP/IP Adapter API Include file cross-reference

TCP/IP Adapter API Filename	VAX-PAMS Equivialent	Comments
p_entry.h	PAMS_C_ENTRY_POINT	Entry point definitions
p_proces.h	PAMS_C_PROCESS	
p_group.h	PAMS_C_GROUP	
p_typecl.h	PAMS_C_TYPE_CLASS	Type and Class definitions
p_return.h and p_aretur.h	PAMS_C_RETURN and PAMS_C_RETURN_DEF	Return code definitions
p_symbol.h	PAMS_C_SYMBOL_DEF	Global symbol definitions
sbsmsgde.h	SBS_MSG_DEF	SBS symbol definitions
availmsg.h	AVAIL_MSG_DEF	
p_tcpip1.h	PAMS_C_TCPIP1	TCP/IP Adapter definitions (contents are identical)
p_tcpip2.h	PAMS_C_TCPIP2	TCP/IP Adapter definitions (contents are identical)

The TCP/IP include files for non-VMS clients are designed to make client applications compatible with the non-VMS versions of *PAMS*, such as *PC-PAMS*. The differences are minor and the include files contain definitions to handle multiple operating systems.

5.2 Differences between PAMS services and the TCP/IP Adapter API

5.2.1 Use of PAMS$DEBUG logical on VMS

The use of **PAMS$DEBUG** currently will not affect the TCP/IP Adapter API, and will not cause any traceback or debugging information to appear for the **PAMS_xxx** calls in the client. Instead, setting the variable *tcpip_debug = TRUE* will cause the TCP/IP API to display its own internal trace and value information to help with debugging.

5.2.2 Additional Status Return Values

Because the TCP/IP Adapter API must support multiple systems and provide communications over a TCP/IP datalink, The TCP/IP API functions have a few additional return status values than those documented for the corresponding *PAMS* service. Also, to provide compatibility with all *PAMS* products, both an **SS$_NORMAL** and a **PAMS__SUCCESS** have been defined for reference.

Table 5–2 lists the status values can be returned in addition to the ones listed in the *PAMS* documentation. These values are possible on ANY TCP/IP API *PAMS* function call.

Table 5–2 Additional Status return values

Symbol	Description
PAMS__PAMSDOWN	Message Bus is inaccessible or the TCP/IP Adapter Process is not available.
PAMS__NOTSUPPORTED	The *PAMS* function referenced is not available with the TCP/IP Adapter API, or the TCP/IP Adapter Process currently running on the host system does not support this release of the TCP/IP API.
PAMS__NETERROR	Unrecoverable TCP/IP communications error encountered
PAMS__SUCCESS	Success. On VMS, this symbol has the same meaning and value as **SS$_NORMAL**
PAMS__BIGBLKSIZ	Attempt to build or send a message larger than the maximum TCP/IP message size allowed

To keep product compatibility, an unrecoverable TCP/IP communications error will result in a return status of **PAMS__PAMSDOWN** or **PAMS__NETERROR**. If there is a need to see the specific error status that resulted; the client program can reference the external integer variable **tcpip_status**, which will contain the returned error value (*errno*) for the socket call.

5.2.3 Other Differences and Restrictions

See Appendix A for a list of the current restrictions and limitations of using the TCP/IP Adapter and API.

5.3 Creating Client Applications

The simplest way to write a TCP/IP Adapter client application is to first write and debug it directly on an existing *PAMS* system, such as *VMS-PAMS* or *PC-PAMS*, and follow that product's documentation. *VMS-PAMS* is a requirement for the TCP/IP Adapter, so it should always be available as a resource. Client applications can then be easily modified to run using the TCP/IP Adapter API, by following the instructions in Section 5.4.

Client's can also be written and tested on non-PAMS systems using the TCP/IP Adapter. The message bus and the TCP/IP Adapter Process will need to be running on the VMS host system first, perhaps interactively with the **PAMS$DEBUG** logical enabled to help with debugging. It is also suggested that the **samp_tcp** program supplied with the product be tested to ensure that the entire TCP/IP Adapter "environment" performs as expected. Clients can then be coded and tested as described in the *PAMS-VMS* documentation, with the exceptions noted in Section 5.4.

5.4 Modifying Client Applications to use the TCP/IP Adapter

The modifications to a client application for use with the TCP/IP Adapter API are minor, and are such that a client application having these modifications will still compile and execute with any *PAMS* package without the need for the TCP/IP Adapter. The changes required fall into two categories:

1. Changes and additions to the *#include* statements.
2. Setting two external variables to contain the *host name* and *port number* of the desired TCP/IP Adapter Process to be used for communications.

This list and the chapter sections that follow detail all the changes necessary to use the TCP/IP API.

1. Be sure to include the **errno.h** and **stdio.h** file at the beginning of the program.
2. Include the TCP/IP Socket definition files. These vary slightly for VMS and non-VMS systems. Example 5–1 shows how to code this for portability with both environments. The code for including these files MUST come before the include files for *PAMS*.
3. Add *#include* statements for the *PAMS* TCP/IP Adapter include files; these must be placed AFTER all other *PAMS* include statements. It is also recommended that the include statements be changed to better support multi-system compatibility, as shown in Example 5–1.

Note: If the client application source was originally coded for use with *PC-PAMS*, some additional changes may be required due to incompatibilities with early releases of that product. Please consult Appendix E for details.

4. Set the values of two external variables; *adapter_port* and **adapter_host*. These variables need to be set BEFORE any TCP/IP Adapter API function is referenced. **adapter_host* needs to be set to the address of a *string* containing the name of the host system that contains the desired the TCP/IP Adapter Process, and *adapter_port* (*unsigned integer*) needs to be set to the port number that TCP/IP Adapter Process is using for socket communications. Example 5–2 shows sample code on how to set these variables by specifying them as arguments during client program activation.

Example 5–1 Recommended C code for TCP/IP and PAMS #include statements

```
#include <errno.h>
#include <stdio.h>

        /* Include TCP/IP Socket library files */

#ifdef VMS
#include <types.h>
#include <socket.h>
#include <in.h>
#include <netdb.h>        /* change to comply with BSD 4.3 */
#include <inet.h>
#include <ucx$inetdef.h>  /* INET Symbol definitions */
#else
#include <sys/types.h>
#include <sys/socket.h>
#include <netinet/in.h>
#include <netdb.h>
#include <arpa/inet.h>
#include <sys/uio.h>
#endif

        /* Include PAMS specific definition files. */

#ifdef VMS
#include pams_c_entry_point       /* PAMS func type declarations */
#include pams_c_process           /* Known Process number def’s. */
#include pams_c_group             /* Known Group number defs     */
#include pams_c_type_class        /* Generic Type/Class defs     */
#include pams_c_return_status_def /* PAMS Func return status defs*/
#include pams_c_symbol_def        /* Generic PSEL/DSEL defs      */

#include pams_c_tcpip1.h          /* TCPIP definitions for PAMS  */
#include pams_c_tcpip2.h          /* TCPIP definitions for PAMS  */

#else

#include "p_entry.h"              /* PAMS func type declarations */
#include "p_proces.h"             /* Known Process number def’s. */
#include "p_group.h"              /* Known Group number defs     */
#include "p_typecl.h"             /* Generic Type/Class defs     */
#include "p_return.h"             /* PAMS Func return status defs*/
#include "p_aretur.h"             /* remaining PAMS status defs  */
#include "p_symbol.h"             /* Generic PSEL/DSEL defs      */

#include "p_tcpip1.h"             /* TCPIP definitions for PAMS  */
#include "p_tcpip2.h"             /* TCPIP definitions for PAMS  */
#endif
```

5 If desired, the external *char* variable *tcpip_debug* can be set to *tcpip_debug = TRUE*. This will cause the TCP/IP Adapter API to print trace and debugging information as it executes. Using this variable is optional. Example 5–3 shows a suggested method of setting this variable, and some sample output appears in Appendix D.

6 Although it may not necessary, it is recommended that all references to **SS$_NORMAL** be changed to **PAMS__SUCCESS**.

7 Recompile the program and link it with the **p_tcpfnc** and **p_tcpip** API objects.

Example 5–2 Setting *host name* and *port number* by arguments

```
main(argc,argv)
int argc;
char **argv;
{
 short msg_area_size = BUF_SIZE+1;

 if (argc != 3 )
 {
     printf("?Host or port specification missing.\n");
     printf("Usage: client hostname portnumber.\n");
     printf("Note: hostname should be in quotes.\n");
     exit();
 }
 else
 {
     if ((adapter_port = atoi(argv[2])) == 0)
     {
  printf("Port number cannot be zero\n");
  exit();
     }
     adapter_host = argv[1];
 }
 /* Examine and use passed parameters for host and port */
```

Example 5–3 Setting *tcpip_debug*

```
main()
{
#ifdef PAMS$TCPIP_ADAPTER
    tcpip_debug = TRUE;
#endif
```

5.5 samp_tcp and tcptest

samp_tcp.c is the **simple_client.c** program, provided with *PAMS* product, modified for use with the TCP/IP Adapter and to be operating system independent. The **tcptest** test program is very similar, except that it performs a communications test to the TCP/IP Adapter process and displays information concerning system configuration and whether the Client operating system and TCP/IP Adapter API are supported. When using the TCP/IP Adapter for the first time, it is suggested that the **tcptest** program be compiled and run before any other client applications. Example 5–4 show a sample output from **tcptest**.

5.6 Compiling and Linking TCP/IP Adapter Clients on non-VMS

The TCP/IP API should not normally require special compile or linking options. Datatypes are explicitly declared in full and the program code is designed not to make assumptions concerning the operating environment. Compile and linking programs vary per system and the user will need to reference the applicable documentation. The **samp_tcp** and **tcptest** sample client applications supplied with the TCP/IP Adapter can be used to test compiling and linking.

Example 5–4 Sample output from tcptest

```
PAMS TCPIP Client test program
Performing test message...

Client V3.0-02, protocol revision 2
Client’s TCP/IP msg size maximum is 32767
Client’s  PAMS  msg size maximum is 32000

Remote Adapter’s information

   Protocol Rev: 2
   Ident string: V3.0-02
   OS ident val: 1
   PSB length  : 32  (ours is 32)
   Retrn status: 1

   Process number is ’600.201’

   Enter message or generate EOF (CTRL/Z on VMS) to exit...

   > HELLO

   CLIENT: Sent Msg to 600.201 class=’1’, type=’-123’
type of psb = 1
call dependant = 0
del_psb_status = 1
seq_number = 13173336, 0
uma_psb_status = 136151171

   CLIENT: Received from 600.201  class=’1’, type=’-123’
           Message=’HELLO’

   Enter message or generate EOF (CTRL/Z on VMS) to exit...
   >

SIMPLE_TCPIP_CLIENT - Done
```

5.7 Compiling and Linking TCP/IP Adapter Clients on VMS

On VMS system, using the TCP/IP Adapter in client applications is a matter of referencing different files in **PAMS$LIB:**. Example 5–5 shows the standard method of compiling and linking the **samp_tcp** program:

Example 5–5 Compiling a TCP/IP Adapter client on VMS

```
$ SET DEF PAMS$EXAMPLES
$ SET DEF [.TCPIP_ADAPTER]
$ CC SAMP_TCP+PAMS$LIB:PAMS_TCPIP.TLB/LIB+PAMS$LIB:PAMS.TLB/LIB
$ LINK SAMP_TCP+PAMS$LIB:PAMS_TCPIP.OPT/OPT
$
```

5.8 Debugging Client Applications

Debugging a client application typically involves including the *tcpip_debug = TRUE* statement in the client, and interactive startup of the TCP/IP Queue Adapter Process on the VMS system with the **PAMS$DEBUG** enabled. The following steps will create a useful debugging environment:

1. Place a *tcpip_debug = TRUE* statement at the beginning of the client application code and rebuild the application.
2. On the host VMS system, perform an interactive startup of the TCP/IP Adapter Process as described in the *TCP/ IP Adapter Process* chapter, making sure to set the **PAMS$DEBUG** logical first.
3. Startup the client application, using whatever debugging tools are available.

As the client application executes, it will display internal TCP/IP Adapter API trace and variable information. As communications occur to the TCP/IP Adapter Process on the VMS host system, *PAMS* trace information will appear there also.

A Restrictions and Limitations

This version of software is has the following restrictios and limitations. These restrictions do not necessarily imply that future versions of the software will have them.

Although all effort has been made to make the code portable and machine independent, this package has not been fully implemented or tested under all platforms. Currently; it has been tested with PAMS V3.0 on VMS 5.1 - 5.3 with UCX V1.2, Ultrix 3.1, and RISC Ultrix 4.0. Testing included use of the VAX C compiler and the Ultrix C compiler. Although the design should work on non-32 bit systems, no compiling or testing on such systems have taken place as of the date of this document. The following list details what restrictions exist for this release

- VMS and ULTRIX have a message size limit of 32k. All other platforms have a maximum size limit of 4k. Although there should be sufficient internal space in the software, it may be possibe under some circumstances to receive a **PAMS__BIGBLKSIZ** error with messages that are below (but near) the maximum size limit. This is because the internal TCP/IP messages also include the PAMS parameters and header information, and the error signifies that the total constructed TCP/IP message exceeded the size maximum
- Only the basic set of the *PAMS* commands have been implemented. See the *Overview* chapter for details.
- *PAMS* AST operations (*PAMS_GET_MSGA*) will not available under this architecture and will return an error.
- Currently, the TCP/IP Adapter Process must be started and already be running before a TCP/IP client application attempts communication.
- Only one client connection is allowed to a specific TCP/IP Adapter Process at a time. If a connection is active and another client requests a connect, the second client's request will be queued and the client will "stall" until the previous connection is closed. Multiple TCP/IP Adapter Processes on the same host system are allowed, provided they use independent port numbers.
- ALL optional *PAMS* arguments in the TCP/IP Adapter API must be specified. If an argument is not desired, a value zero must be passed (this is standard C programming practice, but it is also a requirement for the TCP/IP API to function correctly).
- Only VMS and Ultrix have been tested to date. Further changes to **p_tcpfnc.c** will most likely be required for supporting additional operating systems. This is especially true for non-32 bit systems and/or systems where the internal bytes for integers are stored in a different order than on VAX/VMS.

- **PAMS$DEBUG** currently does not enable a debugging trace mode in the client application. The application must instead set *tcpip_debug = TRUE*.
- Support is provided only for the C programming language. A C language compiler is required for building the client application. For VAX/VMS, the compiler must be *VAX C* V3.0 or greater. *VAX C* is not required on the host system to build the **PAMS_TCPIP_ADAPTER.EXE** image.

B Sample TCP/IP Client Listing

```
/*** module SIMPLE_TCPIP_CLIENT "V3.0-00"
*******************************************************************
*                                                                 *
* Copyright (c) 1990                                              *
* by DIGITAL Equipment Corporation, Maynard, Mass.                *
*                                                                 *
* This software is furnished under a license and may be used and  *
* copied only in accordance with the terms of such license and    *
* with the inclusion of the above copyright notice. This software *
* or any other copies thereof may not be provided or otherwise    *
* made available to any other person. No title to and ownership   *
* of  the  software  is  hereby transferred.                      *
*                                                                 *
* The information in this software is subject to change without   *
* notice and should not be construed as a commitment by DIGITAL   *
* Equipment Corporation.                                          *
*                                                                 *
* DIGITAL assumes no responsibility for the use or reliability of *
*  its software on equipment which is not supplied by DIGITAL.    *
******************************************************************/

/****************************************************************
*                                                               *
* VMS PAMS TCPIP client process.                                *
*                                                               *
* This program provides an example of a very simplistic client  *
* process that essentially performs the following functions:    *
*                                                               *
*                +---------------------------+                  *
*                |  Declare Process to PAMS  |                  *
*                |  using PAMS_DCL_PROCESS   |                  *
*                +---------------------------+                  *
*                              |                                *
*                              V                                *
*                +---------------------------+                  *
*                | Initialize Variables, etc.|                  *
*                +---------------------------+                  *
*            +---------------->|                                *
*            |                 V                                *
*            |   +---------------------------+                  *
*            |   |   Wait for a message to   |                  *
*            |   |     be received using     |                  *
*            |   |       PAMS_GET_MSGW       |                  *
*            |   +---------------------------+                  *
*            |                 |                                *
*            |                 V                                *
*            |   +---------------------------+                  *
*            |   |   Send "canned" response  |                  *
*            |   |  message to client using  |                  *
*            |   |       PAMS_PUT_MSG        |                  *
*            |   +---------------------------+                  *
*            |                 |                                *
*            +---------<-------+                                *
*                                                               *
*                                                               *
* Unlink standard PAMS clients, this client performs the PAMS   *
* functions remotely over a network link. Except for some       *
* additional VMS status return codes possibilities and possible *
* longer response time involved with including network I/O,     *
* these functions should be transparent to the calling ap-      *
* plication. The next diagram show the logic flow:              *
*                                                               *
*                +---------------------------+                  *
*           .-   |   Application program     |                  *
*           |    +---------------------------+                  *
*           |                  |                                *
*           |                  V                                *
*           |    +---------------------------+                  *
*   Client -+    |   PAMS function call      |                  *
*           |    +---------------------------+                  *
*           |    | Reformat parameters into  |                  *
*           |    |       a TCP message       |                  *
*           |    +---------------------------+                  *
*           |    |   Perform TCP to remote   |                  *
*           '-   |  adapter process and wait |                  *
*                +---------------------------+                  *
*                              \                                *
*                              /                                *
*                              V                                *
*                +---------------------------+                  *
*           .-   |    Receive TCP message    |                  *
*           |    +---------------------------+                  *
*           |    | Reformat message into the |                  *
*           |    |      PAMS function call   |                  *
*  TCP/IP   |    +---------------------------+                  *
*  adapter -+    |   Perform PAMS function   |                  *
*           |    | and format results into a |                  *
*           |    |         TCP message       |                  *
*           |    +---------------------------+                  *
*           |    |   Perform TCP to remote   |                  *
*           '-   |       Client and wait     |                  *
*                +---------------------------+                  *
*                              \                                *
*                              /                                *
*                              V                                *
*                +---------------------------+                  *
*           .-   |    Receive TCP message    |                  *
*           |    +---------------------------+                  *
*           |    | Reformat message into the |                  *
*           |    |   PAMS Returned values    |                  *
*           |    +---------------------------+                  *
*   Client -+    |   Return to application   |                  *
*           |    +---------------------------+                  *
*           |                  |                                *
*           |                  V                                *
*           |    +---------------------------+                  *
*           '-   |   Application program     |                  *
*                +---------------------------+                  *
*                                                               *
*                                                               *
* To build the CLIENT, you compile your image and link it with  *
* the P_TCPFNC and P_TCPIP objects (in addition to the C and    *
* TCP/IP socket libraries). DO NOT LINK WITH PAMS LIBRARIES OR  *
* OBJECTS.                                                      *
*                                                               *
****************************************************************/

/*****************************************************************
* Module:    Sample TCPIP Client
*
* Date:      19-Jul-1990
*
* Environment:
*        UCX V1.2 or higher
*        VMS 5.1 or higher
*        Ultrix 3.1 or higher
*
* Function:
*        This program demos and uses a remote PAMS queue via the
*        UCX TCP/IP IPC routines.
*
*
* Restrictions:
*        See documentation.
*
* Revisions:
*
*
*****************************************************************/

        /* Include standard "C" and system definition files. */

#include <errno.h>
#include <stdio.h>       /* Standard I/O definitions */
```

```
#ifdef VMS
#include <types.h>
#include <socket.h>
#include <in.h>
#include <netdb.h>       /* change to comply with BSD 4.3 */
#include <inet.h>
#include <ucx$inetdef.h> /* INET Symbol definitions */
#else
#include <sys/types.h>
#include <sys/socket.h>
#include <netinet/in.h>
#include <netdb.h>
#include <arpa/inet.h>
#include <sys/uio.h>
#endif

            /* Include PAMS specific definition files. */

#ifdef VMS
#include pams_c_entry_point       /* PAMS function types       */
#include pams_c_process           /* Known Process number defs */
#include pams_c_group             /* Known Group number defs   */
#include pams_c_type_class        /* Generic Type/Class defs   */
#include pams_c_return_status_def /* PAMS return status defs   */
#include pams_c_symbol_def        /* Generic PSEL/DSEL defs    */

#include pams_c_tcpip1            /* TCPIP defs for PAMS       */
#include pams_c_tcpip2            /* TCPIP defs for PAMS       */
#else
#include "p_entry.h"              /* PAMS function declarations*/
#include "p_proces.h"             /* Known Process # def’s     */
#include "p_group.h"              /* Known Group number defs   */
#include "p_typecl.h"             /* Generic Type/Class defs   */
#include "p_return.h"             /* PAMS return status defs   */
#include "p_aretur.h"             /* remaining PAMS status defs*/
#include "p_symbol.h"             /* Generic PSEL/DSEL defs    */

#include "p_tcpip1.h"             /* TCPIP defs for PAMS       */
#include "p_tcpip2.h"             /* TCPIP defs for PAMS       */
endif

#define do_forever while (1==1)
#define BUF_SIZE 512               /* Size for local buffers */


        /* Define data type for PAMS target/source addresses. */

typedef union
  {long int      all;
    struct
     {short int  process;
      short int  group;
     } au;
  } pams_addr;


        /* Declare local variables, pointers, and arrays */

static long int   status;            /* Completion status code */
static long int   proc_num_req;      /* Requested process #    */
static pams_addr  proc_num_act;      /* Actual process #       */
struct psb        psb;               /* PAMS status block      */

        /* Define outbound ("put") message variables and arrays */

static pams_addr  source;            /* Source queue address   */
static char       put_buffer[BUF_SIZE+1];
                                     /* Message buf+1 null byte*/
static short int  put_class;         /* Message class code     */
static char       put_delivery;      /* Delivery mode          */
static short int  put_priority;      /* Message priority       */
static long int   put_resp_que;      /* Response queue         */
static short int  put_size;          /* Message size           */
static long int   put_timeout;       /* Timeout for blocked msg*/
static short int  put_type;          /* Message type code      */
static char       put_uma;           /* UMA                    */

        /* Define inbound ("get") message variables and arrays */
```

```
static pams_addr  target;            /* Target queue address   */
static char       get_buffer[BUF_SIZE+1];
                                     /* Message buf+1 null byte*/
static short int  get_class;         /* Message class code     */
static short int  get_priority;      /* Message priority       */
static long int   get_select;        /* Message selection mask */
static short int  get_size;          /* Message size           */
static long int   get_timeout;       /* Timeout for blocked msg*/
static short int  get_type;          /* Message type code      */


main(argc,argv)
int  argc;
char **argv;
{
        short  msg_area_size = BUF_SIZE+1;

        if (argc != 3 )
        {
            printf("?Host or port specification missing.\n");
            printf("Usage: client hostname portnumber.\n");
            printf("Note: hostname should be in quotes.\n");
            exit();
        }
        else
        {
            if ((adapter_port = atoi(argv[2])) == 0)
            {
                printf("Port number cannot be zero\n");
                exit();
            }
            adapter_host = argv[1];
        }
        /* Examine and use passed parameters for host and port */

             /**********************************************/
             /*          Initialization Section            */
             /*                          - - - - -         */
             /* Declare our process to PAMS and initialize */
             /**********************************************/

   printf ("\n\nTCPIP Client Process Example\n");

   /*  Call pams_dcl_process to attach to ourself to the PAMS  */
   /*  message bus, and assign our process (queue) number.     */

   proc_num_req = 0;
   status = pams_dcl_process (&proc_num_req, &proc_num_act,
       0,0,0,0);

   if (status != PAMS__SUCCESS)
      {
        printf ("Error returned by PAMS_DCL_PROCESS ’%d’\n",
              status);
        exit (status);
        }


   printf ("\n Process number is ’%d.%d’\n\n",
       proc_num_act.au.group,
       proc_num_act.au.process);


   /* Init variables that we will be using for messaging. */

   get_priority = 0;                /* Receive all messages */
   get_timeout  = 600;              /* 60 sec timeout on rcv*/
   get_select   = 0;                /* No special selection */
   put_class    = 1;                /* Send class 1 message */
   put_priority = 0;                /* Send at standard pri */
   put_resp_que = 0;                /* Response que; default*/
   put_type     = -123;             /* Msg type user defined*/
   put_timeout  = 0;                /* Accept standard tmo  */
   put_delivery = PDEL_MODE_NN_MEM; /* No notify, memory que*/
   put_uma      = PDEL_UMA_DISC;    /* Discard UMA          */

   target.all = proc_num_act.all;   /* send to ourselves */
```

```
do_forever
{
/* Clear type-ahead buffer, then prompt for and read the
   message. If <CTRL-Z> pressed (EOF), we are done, break
   out of the loop. */

   fflush (stdin);
   printf ("\n\nEnter message or generate an EOF to exit...\n   >");
   if (gets(put_buffer) == ’\0’) break;

   /* Set put message size to buffer length +1 */

   put_size = strlen(put_buffer)+1;

   if (put_size <= msg_area_size)
   {
   /* Send the message to the target process queue.
      If an error is returned, report it and exit,
      otherwise display class & type. */


         status = pams_put_msg (put_buffer, &put_priority,
                          &target.all,
                          &put_class, &put_type,
                          &put_delivery,
                          &put_size,
                          &put_timeout, &psb, &put_uma,
                          &put_resp_que);

         if (status != PAMS__SUCCESS)
         {
             printf ("Error returned by PAMS_PUT_MSG code=%d\n",
                  status);
             exit (status);
         }

         printf ("\n  CLIENT: Sent Msg to %d.%d class=’%d’,
                     type=’%d’\n",
             target.au.group, target.au.process,
             put_class, put_type);

        /* Now wait for the adapter process to send the reply
           If timeout error, report it and continue. If other
           error, report it and exit.  Else, display */

         status = pams_get_msgw (get_buffer, &get_priority,
                           &source.all,
                           &get_class, &get_type,
                           &msg_area_size,
                           &get_size,
                           &get_timeout, &get_select, &psb,
                           0, 0, 0, 0);

         if (status == PAMS__TIMEOUT)
             printf ("PAMS_GET_MSGW Timeout\n");

         else if (status != SS$_NORMAL)
         {
             printf ("Error returned by PAMS_GET_MSGW code=%d\n",
                     status);
             exit (status);
         }
         else
         {
           printf("\nCLIENT: Received from %d.%d class=’%d’, type=’%d’\n",
               source.au.group, source.au.process,
               get_class, get_type);
           printf ("           Message=’%s’\n", get_buffer);
         }
    }
    else
         printf("input exceeds buffer max of %d charactes\n",
             msg_area_size);
}
             /*  End of Main Processing Loop  */

   /************************************************/
   /*                 EXIT Routine                 */
   /*                   - - - -                    */
   /*  User pressed <CTRL-Z>.  Clean-up and exit.  */
   /************************************************/
```

```
      /* If the adapter process was started, then use the
         system() function to stop the adapter process using
         the DCL STOP command. */

      /* Tell the user we’re done, then exit. */

   status = pams_exit();

   printf ("\n\nSIMPLE_TCPIP_CLIENT - Done\n");
   exit (SS$_NORMAL);
}
```

C Sample Listing of PAMS_TCPIP.OPT

The following is a copy of the options file for linking VMS client applications with the TCP/IP Adapter API.

```
!  +---------------------------------------------+
!  | General Purpose VAX PAMS TCP/IP Adapter     |
!  |          Linker Options File                |
!  |                                             |
!  | File:  PAMS$LIB:PAMS_TCPIP.OPT              |
!  +---------------------------------------------+
!
     PAMS$LIB:PAMS_TCPIP.OLB/LIB
     SYS$LIBRARY:UCX$IPC.OLB/LIB
     SYS$SHARE:VAXCRTL.EXE/SHARE
!
!----------- end of PAMS_TCPIP.OPT --------------------
!
```

D Sample tcpip_debug Output

The following is an example of output produced by setting *tcpip_debug = TRUE*.

```
SIMPLE_TCPIP_CLIENT - SIMPLE TCPIP Client Process Example

PAMS_TCPIP: Entering TCPIP_SEND routine
PAMS_TCPIP: message sent, status/bytecount = 34

PAMS_TCPIP: Entering TCPIP_RECEIVE; awaiting msg...
PAMS_TCPIP: received msg from client; packetsize = 26
PAMS_TCPIP: received msg of 26 bytes

   Process number is '600.202'

   Enter message or generate an EOF (CTRL/Z on VMS) to exit...
   > hello

PAMS_TCPIP: Entering TCPIP_SEND routine
PAMS_TCPIP: message sent, status/bytecount = 33

PAMS_TCPIP: Entering TCPIP_RECEIVE; awaiting msg...
PAMS_TCPIP: received msg from client; packetsize = 44
PAMS_TCPIP: received msg of 44 bytes

   CLIENT: Sent Msg to 600.202 class='1', type='-123'

PAMS_TCPIP: Entering TCPIP_SEND routine
PAMS_TCPIP: message sent, status/bytecount = 25

PAMS_TCPIP: Entering TCPIP_RECEIVE; awaiting msg...
PAMS_TCPIP: received msg from client; packetsize = 60
PAMS_TCPIP: received msg of 60 bytes

   CLIENT: Received from 600.202  class='1', type='-123'
           Message='hello'

   Enter message or generate an EOF (CTRL/Z on VMS) to exit...
   >

PAMS_TCPIP: Entering TCPIP_SEND routine
PAMS_TCPIP: message sent, status/bytecount = 6

PAMS_TCPIP: Entering TCPIP_RECEIVE; awaiting msg...
PAMS_TCPIP: received msg from client; packetsize = 10
PAMS_TCPIP: received msg of 10 bytes

PAMS_TCPIP: Entering TCPIP_SHUTDOWN routine

SIMPLE_TCPIP_CLIENT - Done
```

E PC-PAMS Compatibility

Early releases of *PC-PAMS* had a different **PSB** and other definitions than all other implementations of *PAMS*. The include files provided with the TCP/IP Queue Adapter are "supersets" and should be used in place of the files provided in the *PC-PAMS* releases. These new include files are capable of generating either type of structures, depending on the setting of the **GEN_PAMS_V1** definition in the file. **GEN_PAMS_V1** defaults to building the compatible structures.

Client applications written for early releases of *PC-PAMS* will need modification for use on other operating systems or with the TCP/IP Adapter. The changes concern modifying the variable names for the elements of the *psb* and others to match those defined in the include file.

F Using the TCP/IP Adapter under PAMS V2.5

The TCP/IP Adapter process will support connections from clients designed for *PAMS V2.5 or 3.0*. The Adapter Process must be built against the highest version of *PAMS* that it will have to support in a client application. In other words; if the Adapter process will only be connecting to V2.5 clients, then the Adapter Process can be built against *PAMS V2.5* or *PAMS V3.0*. If the Adapter Process must support both V2.5 clients **AND** V3.0 clients, then the Adapter process must be build against *PAMS V3.0*.

On the client side; the *p_tcpip1.h* file will require a modification and the Client applications and the *p_tcpfnc* and *p_tcpip* modules will need to be re-compiled. The modification is to "un-comment" the definition for the "PAMSV25" symbol near the beginning of the file. After that, re-compile *p_tcpip* and *p_tcpfnc*. If the clients Applications are to run on the VMS host system, then *p_tcpip1.h* will have to be re-inserted in the PAMS$TCPIP.TLB library as *pams_c_tcpip1*, and the *p_tcpip* and *p_tcpfnc* modules will have to be re-inserted into the PAMS$LIB:PAMS_TCPIP.OLB object library. Finally, re-compile and re-link the Client Application program(s).

Index