HEWLETT PACKARD

HP 9000
Computers

DTC Device File Access Utilities and Telnet Port Identification

HP 9000 Computers

DTC Device File Access Utilities and Telnet Port Identification

Edition 1
E0992

B1014-90012
Printed in U.S.A. 09/92

Notice

Hewlett-Packard makes no warranty of any kind with regard to this material, including, but not limited to, the implied warranties of merchantability and fitness for a particular purpose. Hewlett-Packard shall not be liable for errors contained herein or for incidental or consequential damages in connection with the furnishing, performance, or use of this material.

Hewlett-Packard assumes no responsibility for the use or reliability of its software on equipment that is not furnished by Hewlett-Packard.

RESTRICTED RIGHTS LEGEND

Use, duplication or disclosure by the U.S. Government is subject to restrictions as set forth in subparagraph (c)(1)(ii) of the Rights in Technical Data and Computer Software clause at DFARS 252.227-7013 for DoD agencies, and subparagraphs (c)(1) and (c)(2) of the Commercial Computer Software Restricted Rights clause at FAR 52.227-19 for other agencies.

Hewlett-Packard Co.
19420 Homestead Rd.
Cupertino, CA 95014 U.S.A.

Printing History

New editions are complete revisions of the manual. Update packages, which are issued between editions, contain additional and replacement pages to be merged into the manual by the customer. The dates on the title page change only when a new edition or a new update is published. No information is incorporated into a reprinting unless it appears as a prior update; the edition does not change when an update is incorporated.

Note that many product updates and fixes do not require manual changes and, conversely, manual corrections may be done without accompanying product changes. Therefore, do not expect a one-to-one correspondence between product updates and manual updates.

Edition 1 . . . . . September 1992

Contents

Figures

Tables

1

Introduction

This manual addresses the topic of access to Datacommunication and Terminal Controller (DTC) serial devices on HP 9000s via pseudoterminal (pty) device files. The focus of this topic are the DTC Device File Access Utilities and enhancements to the ARPA/9000 Telnet service known as Telnet Port Identification. In addition, there is general discussion on the topic of pty use with HP DTCs.

Purpose

This manual is intended to fulfill these purposes:

- To provide information regarding the configuration, verification, and use of the *DTC Device File Access Utilities.*
- To provide information regarding the configuration, verification, and use of Port Identification enhancements of the ARPA/9000 Telnet service.
- To provide information on when to use, and how to apply these tools for accessing pty device files associated with devices on a DTC.

Intended Audience

This manual is intended for at least three types of users:

- The HP 9000 system administrator or network administrator who defines and configures the device files associated with DTC devices on the system.
- The HP 9000 system operator or network operator who may implement the actual tasks set up by the system or network administrator.
- The applications programmer who needs programmatic control of devices on the DTC using standard HP-UX input/output calls, such as **open**, **close**, **read**, and **write**. This programmer may also have applications using devices connected to MUX ports and wishes to extend the application to use DTC devices.

Prerequisites

Before reading this manual and using this software, you should have the following background:

- Familiarity with HP-UX operating system and system administration, especially devices and device files.
- Familiarity with configuring a DTC with either the DTC Manager/UX product (HP J2120A) or the OpenView DTC Manager product on the PC (HP D2355A).
- Familiarity with programming devices and understanding device files for pseudoterminals (ptys) or MUXes.

Supported Configurations

There are several types of software and hardware products and services to consider before using DTC Device File Access utilities. Refer to the "Related HP Documentation" section later in this chapter to obtain more information on these products and services.

- **HP-UX operating system version** 8.0 or 9.0. The DTC Device File Access Utilities are included with HP-UX version 9.0. The DDFA Utilities consists of executable files, default configuration files, and manual reference pages.

  An electronic version of DDFA can be obtained for HP-UX version 8.0. Contact your HP representative for more information.

- **ARPA/9000 services.** DDFA is an extension of the Telnet service. Because Telnet is part of ARPA/9000 and DDFA uses the telnet protocol, DDFA is automatically installed with ARPA/9000. Therefore, the system must have ARPA/9000 and the LAN Link configured and operating properly.

- **HP 9000 Series 300, 400, 700, and 800 systems.** The DDFA Utilities are supported on HP 9000 Series 300, 400, 700, and 800 systems. Because all these systems have ARPA/9000 as part of their HP-UX system and DDFA is part of Telnet, these systems also have DDFA installed.

- **DTC Manager/UX or OpenView DTC Manager.** The HP DTCs installed on an HP 9000 Series 800 are configured and managed by the *HP DTC Manager/UX* (HP J2120A) or the PC-based *OpenView DTC Manager* (HP D2355A).

  The PC-based OpenView DTC Manager also configures and manages DTCs installed on HP 9000 Series 300, 400, and 700 systems and on HP 3000 systems.

The version for DTC Manager/UX should be A.02.00 or later. The version for the OpenView DTC Manager should be 12.1 or later.

- **Printers and plotters.** DDFA Utilities can be used with the HP-UX spooler for printers and plotters for which a model script exists. Printers and plotters supported on HP DTCs can also be used with DDFA.
- **Terminals.** Any terminal supported for use with HP DTCs is also supported.
- **Modems.** Any modem supported for use with HP DTCs is also supported.
- **HP DTCs.** The following DTCs are supported on HP 9000 systems:
  - HP 2340A - DTC 16 with 16 ports.
  - HP 2345A - DTC 48 with 48 ports.

Manual Organization

This manual is outlined as follows:

Chapter 1 — Introduction. This chapter gives an overview of the software structure and related products.

Chapter 2 — Overview of DTC Device File Access Utilities. This chapter explains the benefits and features of using DTC Device File Access utilities. Summaries of the DDFA software installation and configuration steps are also provided.

Chapter 3 — Telnet Port Identification. This chapter explains how incoming connections from the DTC to the system are identified and handled by the Telnet service.

Chapter 4 — HP-UX Access to DTC Devices. This chapter explains how MUXes and DTCs handle device files differently. The chapter also explains the differences between incoming and outgoing connections and how they are handled by DTC Device File Access utilities and the Telnet port identification features.

Chapter 5 — DDFA Configuration. This chapter explains the configuration files and process of the DDFA Utilities in more detail.

Chapter 6 — Application Examples. This chapter presents some common situations for using special device files for DTCs, such as the HP-UX printer spooler and programming applications.

Chapter 7 — Simple Troubleshooting. This chapter explains areas most likely to cause problems for outgoing and incoming connections.

Appendix A — Error Messages for dpp. This appendix lists and explains the error messages from the dedicated port parser, **dpp**.

Appendix B — Migrating from Device Access/ARPA Software. This appendix explains how to migrate from the Device Access/ARPA product to DDFA.

Appendix C — DDFA Manual Reference Pages. This appendix is a printout of the DDFA manual reference pages (manpages). Also check the online manual reference pages on your system.

Related HP Documentation

HP-UX System manuals:

- *System Administration Tasks* (the HP part number is different for each of the HP 9000 systems)
- *Installing and Updating HP-UX* (the HP part number is different for each of the HP 9000 systems)

LAN and ARPA manuals:

- *Administering ARPA Services* (B1014-90008)
- *Installing and Administering LAN/9000* (98194-60530)

DTC hardware manuals:

- *HP 2345A Datacommunications and Terminal Controller Installation and Service Manual* (02345-90021) for the DTC 48 product.
- *HP 2340A Datacommunications and Terminal Controller Installation and Service Manual* (02340-90001) for the DTC 16 product.

DTC Manager manuals:

- *Using the OpenView DTC Manager* (D2355-90001) for the PC-based DTC manager.
- *Using the HP OpenView DTC Manager/UX* (J2120-62000) for the host-based DTC manager on an HP 9000 Series 800.

2

Overview of DTC Device File Access Utilities

This chapter gives a brief overview about DTC Device File Access (DDFA) Utilities. Use this chapter to get started quickly by using the summaries of installation and configuration steps. Chapters 3, 4, and 5 provide more detailed information on DDFA.

How DDFA Utilities Work

The DTC Device File Access Utilities are a group of configuration files, executable files, and a daemon. Together these utilities let the HP 9000 system administrator predefine a set of ptys to communicate with asynchronous devices connected to the HP Datacommunications and Terminal Controller (DTC). Previously, ptys were assigned by the system at random to devices on a DTC. By having specific pty assignments, these utilities provide both the HP-UX spooler and HP-UX user applications with a means of accessing asynchronous devices on DTCs.

Figure 2-1 illustrates how devices on a DTC are assigned from a pty pool (without the DDFA Utilities).

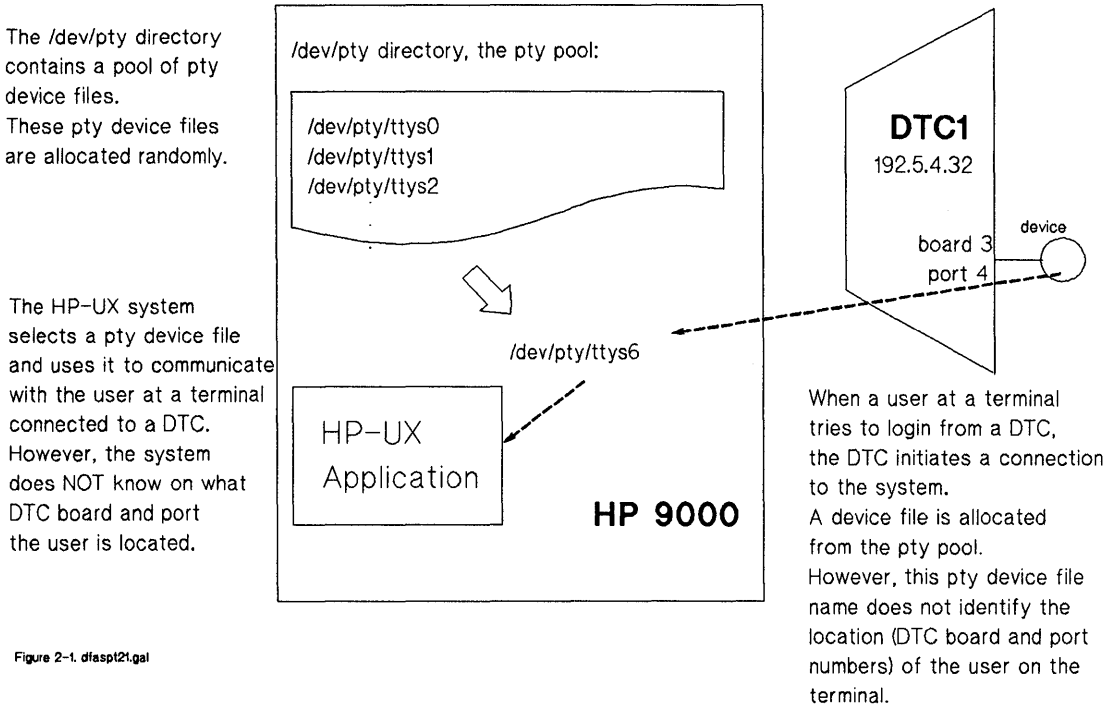


Figure 2-1. dfaspt21.gal

Figure 2-1. How devices on a DTC are assigned from the pty pool

DDFA includes the following configuration files, executable files, and daemon. These items are explained briefly below and in more detail in the other chapters of this manual. Refer also to the online manual reference pages (man pages) for these DDFA Utilities. These man pages are printed in Appendix C.

dp — Dedicated Port configuration file (**/etc/newconfig/ddfa/dp**). This is a master template text file containing information for each configured DTC port. Each line specifies the IP address, board, port, pty device file name, and port configuration file name (**/etc/ddfa/newconfig/pcf** or **etc/ddfa/pcf**; see **pcf** below) for each configured DTC port. DDFA uses this information to setup and to manage a connection for each specified DTC port.

The template file for the dedicated port configuration file is in **/etc/newconfig/ddfa/dp**. You should copy it into **/etc/ddfa/dp** and alter it there.

`pcf` — Port Configuration File (**/etc/newconfig/ddfa/pcf**). This a master template text file containing default port configuration information. It is copied and altered as needed for each device on each port. The template file for the port configuration file is in **/etc/newconfig/ddfa/pcf**. You should copy it into **/etc/ddfa/pcf** and alter it there. The **pcf** is referenced in the **dp** file.

`dpp` — Dedicated Port Parser (**/etc/dpp**). The Dedicated Port Parser (**/etc/dpp**) is an executable file. It parses the **/etc/ddfa/dp** file. Based on the contents of the **dp** file, the **dpp** spawns an Outbound Connection Daemon (**/etc/ocd**) for each outbound entry specified in the **dp** file. The **dpp** can be run from the Shell or it can be included in **/etc/netlinkrc** or **/etc/rc** to automatically run the DDFA software each time the system is booted. Network related tasks are usually put into **/etc/netlinkrc**.

`ocd` — Outbound Connection Daemon (**/etc/ocd**). This daemon manages the connection and data transfer to each DTC port. Normally, **ocd** is spawned by the **dpp**. However, **ocd** can also be run from the Shell with all the parameters from the **dp** file specified on the command line.

`ocdebug` — Outbound Connection Daemon in debug mode (**/etc/ocdebug**). This daemon is a special debugging version of **ocd** and performs the same tasks as **ocd**. In addition, the **ocdebug** daemon logs debug messages to a log file, **/usr/adm/ocd<pid>**, for troubleshooting purposes. The **ocdebug** daemon is explained in Chapter 7, "Simple Troubleshooting."

A daemon is created for each configured port based on the information in a DDFA configuration file called the dedicated port file. This file is in **/etc/ddfa/dp**. When the daemon is spawned, it takes a pty from the pty pool in the **/dev** directory. The daemon then creates a device file with the same major and minor number as the pty slave and gives it a new name as specified in the **dp** file. The new device file is known

as the "pseudonym." User applications are to use this pseudonym or pty device file name to access the DTC port using standard HP-UX intrinsics (such as **open**, **close**, **read**, and **write**).

The daemon listens on the pty, until an application does an **open** call using the pseudonym (**open(<pseudonym>)**). Then the daemon manages the connection to the DTC port. The result is that the DTC port is addressed by a *predefined* device file name. All this activity is transparent when you use a terminal or printer on the DTC. For example, the **lpadmin** command can be used to configure a printer on a DTC port by using the pseudonym.

Another term for the pseudonym is a "well-known pty" or "well-known device file name." The name is "well-known" because it is documented or configured in an accessible location. For DDFA, the well-known pty device file names or pseudonyms are listed in the dedicated port file, **/etc/ddfa/dp**.

Benefits and Features

The DTC Device File Access Utilities provide the following benefits:

- ***Accessing DTC devices is like accessing MUX devices.*** DDFA Utilities allow the system administrator to set up a correspondence between DTC ports and HP-UX device files. With this correspondence defined, the system spooler or a user application can manipulate well-known device files to read and write to specific DTC ports.
- ***Configuring the printer spooler for printers on a DTC can be done with SAM.*** After the correspondence between DTC printers and HP-UX device files has been set-up, using SAM (System Administration Management tool) to configure the spooler using DTC printers is almost identical to using MUX-connected printers. The only difference is that the pty device file name of the DTC printer must be used instead of a tty name for a MUX printer. Moreover, the standard spooler model scripts can be used with the DTC printer(s).
- ***Programming user applications to access DTC devices is done with standard HP-UX system calls.*** After the correspondence between DTC devices and HP-UX device files has been set up, user applications can use the standard HP-UX **read()**, **write()**, **open()**, **close()**, and **ioctl()** calls. These calls access DTC devices by manipulating their corresponding device files.
- ***Specifying device file names in user applications can be for either a MUX or a DTC device.*** When using DDFA Utilities, most applications designed to read or write to MUX-connected devices can read and write to DTC-connected devices without the

need to modify the application. It is only necessary to use the new device file of a DTC device instead of the MUX device file when configuring the application.

All of the above features depend on the host accessing a DTC port based upon a predefined pty device file name. Note that what is really being provided here is a means of opening an outgoing connection from the host to the DTC port by using a pty device file.

DDFA Software Installation

DDFA is automatically installed when your system is updated and ARPA/9000 is installed. There are three basic steps to installing and executing DDFA:

1. Execute the **update** utility on your system to install HP-UX and ARPA/9000. Refer to the *Installing and Updating HP-UX, HP-UX System Administration Tasks Manual,* and *Administering ARPA/9000 Services Manual* for installation information.

   After updating HP-UX, the following DDFA filesets must be on your system:

   ARPA-AUX DTC Device File Access – DDFA

   /etc/dpp
   /etc/ocd
   /etc/ocdebug
   /etc/newconfig/ddfa/dp
   /etc/newconfig/ddfa/pcf

   ARPA-RUN Telnet Port Identification

   /etc/telnetd

   This is just one of the many regular files of the ARPA-RUN fileset. Only telnetd is modified for Telnet Port Identification.

   ARPA-AUX-MAN DTC Device File Access Reference

   /usr/contrib/man/man1m/dpp.1m
   /usr/contrib/man/man1m/ocd.1m
   /usr/contrib/man/man1m/ocdebug.1m
   /usr/contrib/man/man4/dp.4
   /usr/contrib/man/man4/pcf.4
   /usr/contrib/man/man7/ddfa.7

If the DDFA filesets are not present in the **/etc/filesets** directory, then DDFA has not been installed. Refer to the above HP-UX and ARPA manuals for **update** information.

2. Configure the dedicated port (**dp**) files of DDFA. Refer to the next section, "Summary of DDFA Configuration Steps" and Chapter 5, "DDFA Configuration."

3. Configure the board and ports of the DTC(s). The DTC must be configured and downloaded before actually making a connection. Refer to your DTC manager manual, either *Using the DTC Manager/UX Manual* or *Using the OpenView DTC Manager Manual.*

Steps 2 and 3 can be done in either order. The important point is that the DTC must be configured and downloaded before a connection is actually made.

Summary of DDFA Configuration Steps

You must be root or superuser to follow these DDFA configuration steps.

HP recommends keeping a backup of your HP-UX software in order to recover from potential problems in the future. In addition, be sure to read any text README files or README documents supplied with your system.

Configuration for DDFA centers around the **/etc/newconfig/ddfa/dp** file. The following steps summarize the DDFA configuration:

1. Create a directory for the DDFA files. HP recommends **/etc/ddfa**:

```
# mkdir /etc/ddfa
```

2. Copy the master template dedicated port file, **dp**, to the DDFA directory:

```
# cp /etc/newconfig/ddfa/dp /etc/ddfa/dp
```

   Do not alter **/etc/newconfig/ddfa/dp**, so that you keep a master template **dp** file. Altering **/etc/ddfa/dp** is explained in Steps 4 and 5 below.

3. Copy the master template port configuration file, **pcf**, to the DDFA directory:

```
# cp /etc/newconfig/ddfa/pcf /etc/ddfa/pcf
```

   Do not alter **/etc/newconfig/ddfa/pcf**, so that you keep a master template **pcf** file.

4. For each DTC output device (such as a printer) that you wish to permanently associate with a well known device file, perform the following steps:

   a. Determine the IP Address of the DTC, the board number, and port number on the DTC to which the device is connected.

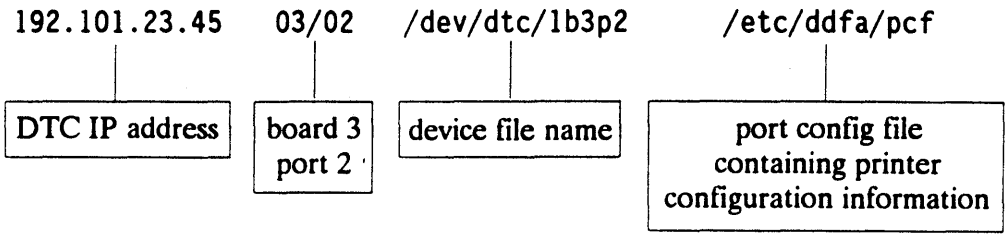
   b. Define a device file name that you will use for this output device.

   c. Create an entry (for each output device) in the **/etc/ddfa/dp** file of the following form:

```
<DTC IP address> <board>/<port> <device file name> <port config file>
```

   The slash (/) must separate the board and port parameters.

   For example, a printer is on board 3 port 2 of a DTC whose IP address is 192.101.23.45. You wish to refer to this printer as **/dev/dtc1b3p2**. The entry in the **dp** file would be:



   Each output device must have a **pcf** file associated to it. Several devices can have the same **pcf** file, or they can each have a unique **pcf** file. Usually, the default **pcf** file is sufficient for most applications or systems.

5. For each DTC input device (such as a terminal) that you wish to permanently associate with a well known device file, perform the following steps. These steps apply to the Telnet Port Identification feature explained in Chapter 3.

   a. Determine the IP Address of the DTC, the board number, and port number on the DTC to which the terminal is connected.

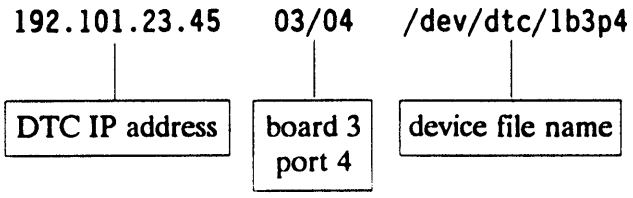
   b. Define a device file name that you will use for this input device.

   c. Create an entry (for each input device) in the **dp** file of the following form:

```
<DTC IP address> <board>/<port> <device file name>
```

The slash (/) must separate the board and port parameters.

For example, a terminal is on board 3 port 4 of a DTC whose IP address is 192.101.23.45. You wish to refer to this terminal as **/dev/dtc1b3p4**. The entry in the **dp** file would be:



Note that a reference to a port configuration file (**pcf**) is not necessary (as shown in Step 4) for a DTC input device such as a terminal.

6. After you have finished editing the **/etc/ddfa/dp** file, execute the dedicated port parser (**dpp**) to scan the **dp** file and start up the **ocd** daemons:

```
# /etc/dpp /etc/ddfa/dp -k
```

7. To configure a DTC printer with the HP-UX spooler, proceed as for a MUX printer, except use the name of the device file that you defined in the **/etc/ddfa/dp** file. Use the standard system model scripts. Refer to the *HP-UX System Administration Tasks Manual* for information on the HP-UX printer spooler.

8. Type the HP-UX command, "**man ddfa**", for more detailed DDFA configuration information.

3

Telnet Port Identification

This chapter briefly explains the enhancements to the Telnet protocol known as **Telnet Port Identification**. More details and its relationship with the DDFA **dp** file are explained in Chapter 4.

Identifying a DTC Port to the System

Telnet Port Identification refers to the ability of the DTC to pass port identification information (board and port numbers) to the host at connection establishment time. This information can be used by the host to map incoming connections to specific pty device files, thereby providing a "dedicated port". In other words, port identification is a means of assigning incoming Telnet connections to predefined pty device files, usually so the identity of the caller can be determined.

Port identification is made possible by a set of enhancements made to the ARPA/9000 Telnet daemon (**telnetd**). Previously with earlier versions of ARPA Telnet, incoming Telnet connections were always assigned pty device files on a random basis. Refer to Figure 2-1 in Chapter 2 for an illustration of random pty device file assignments. Therefore, the common practice of using the device file name to identify the port belonging to a login session had been impossible with DTC-based Telnet connections.

Now, ARPA Telnet allows the system administrator to set up predefined or "well-known" pty names defined in the DDFA dedicated port file, **dp**. In addition, the DTC download code was enhanced so that it passes board and port information to the host (via Telnet) at connection establishment time. Like DDFA Utilities, the **dp file** provides the mappings of the pty names and DTC boards and ports. Internally, Telnet uses a binary lookup file created by **dpp** when **dpp** parses the **dp** file.

Benefits and Features

The port identification enhancements provide one simple, yet highly useful, capability to the system administrator:

- Telnet Port Identification lets the system administrator insure that incoming Telnet connections from specific DTC ports are assigned specific pty device files.

First, the system administrator defines mappings between DTC ports and pty names in the **dp** file using the DDFA Utilities. Then the uses the defined ptys, rather than random ones, for incoming connections from any DTC ports in the **dp** file. Therefore, the identity of a calling DTC port can be determined, or the port can be accessed based upon its well-known pty device file name.

DDFA Utilities and Telnet Port Identification *cannot* be used simultaneously on the same device file. They provide separate functionality. However, they may be used on the system at the same time.

The DDFA Utilities pertain to connections originating from the host. These connections are called *outgoing* or *outbound connections*. Telnet Port Identification pertains to connections originating from the DTC. These connections are called *incoming connections*. Refer to the figures in the next chapter for illustrations of these connections.

4

HP-UX Access to DTC Devices

This chapter explains how HP-UX and DTC connections interact. Also explained are the differences between MUX and DTC connections. Read this chapter to obtain information on why HP-UX access to DTC devices differs from a MUX.

HP-UX access to and from DTC devices is actually provided by means of Telnet connections. HP-UX access with devices connected to a MUX is by means of the MUX driver. To a user logging onto a terminal from either a MUX or a DTC, the terminal functions the same way. The difference occurs internally. The difference is the way in which the device file is assigned to the connection.

Devices connected to a MUX are assigned to tty device file names. Devices connected to a DTC are assigned to *random* pty device file names.

Recall that a device file is an HP-UX file that "points" to a system device. Often, the system administrator uses the name of the device file when configuring software to access that device.

How MUXes and DTCs Handle Device Files Differently

The HP-UX System Administrator creates device files for each of the MUX ports using the HP-UX **insf** or **mknod** command. Each device file created belongs to a specific physical MUX port and device files are named, by convention, so that each identifies a unique MUX port. For example, **/dev/tty2p3** means port 3 on MUX card 2).

A device connected to a DTC communicates with the system via Telnet. Therefore, it is considered to be a logical device, and it is serviced by a **pseudoterminal device driver** (pty). The device is referenced by using the name of the pty device file. Usually, this pty is assigned to the Telnet connection randomly from a pool of free ptys in the **/dev** directory at connection establishment time.

In many cases, the randomness of pty assignments for Telnet users is acceptable, because the physical location of the Telnet user is unimportant. In fact, users of

system-to-system Telnet have always been subject to this situation. However, when a specific DTC device must always be associated with the same pty, then the randomness of pty assignments becomes unacceptable.

Outgoing Connections Without DDFA

An outgoing connection is one in which the *host initiates* the connection to the DTC. For example, a connection between the system and a printer is an outgoing connection.

If an application on the host needs to access a MUX device, the application can read and write to the tty device file that belongs to the MUX device. However, if an application wants to open a DTC device, difficulties arise in finding an appropriate device file to access. The DTC device has been assigned to a random pty device file name.

Normally, it would be impossible to find a device file that uniquely refers to a DTC device. In fact, before an application has accessed a DTC port, no connection to that port even exists. Hence, no pty device file for the device has been assigned. Moreover, there is no easy way for an application itself to establish an outgoing connection to a DTC port via Telnet. There is no interface for an application to initiate a Telnet connection.

Previously, the Device Access/ARPA (DA/AP) product was used to access DTC devices. The Device Access/ARPA software allowed the HP-UX spooler to send output to DTC printers. However, Hewlett-Packard recommends that all customers using Device Access/ARPA migrate to DDFA if they are running HP-UX version 8.0 or later. Refer to Appendix A for information about migrating from Device Access/ARPA to DDFA.

How DTC Device File Access Utilities Help

The DDFA Utilities allow the system administrator to set up a correspondence between pty device files and physical DTC ports. With this correspondence in place, the DTC's pty devices can be used in the same way that MUX-connected tty devices can. This is because the DDFA software eliminates the random assignment of ptys to DTC devices allowing the user or administrator to reference specific DTC devices by well-known or predefined pty names. When an application opens such a device file, the DDFA Utilities transparently manage the establishment of a Telnet connection to the associated DTC port.

Refer to Figure 4-1 for an illustration of how the system, the DTC, and the DDFA Utilities interact with one another.

An outgoing dedicated port is defined in the dp file where a pty device file name is created for each dedicated port (in column 3).

The dedicated port parser (dpp) spawns an outgoing connection daemon (ocd) for each outgoing dedicated port.

When a host application accesses the pty device file specified in the dp file, a connection to the DTC device is initiated and communication can begin.

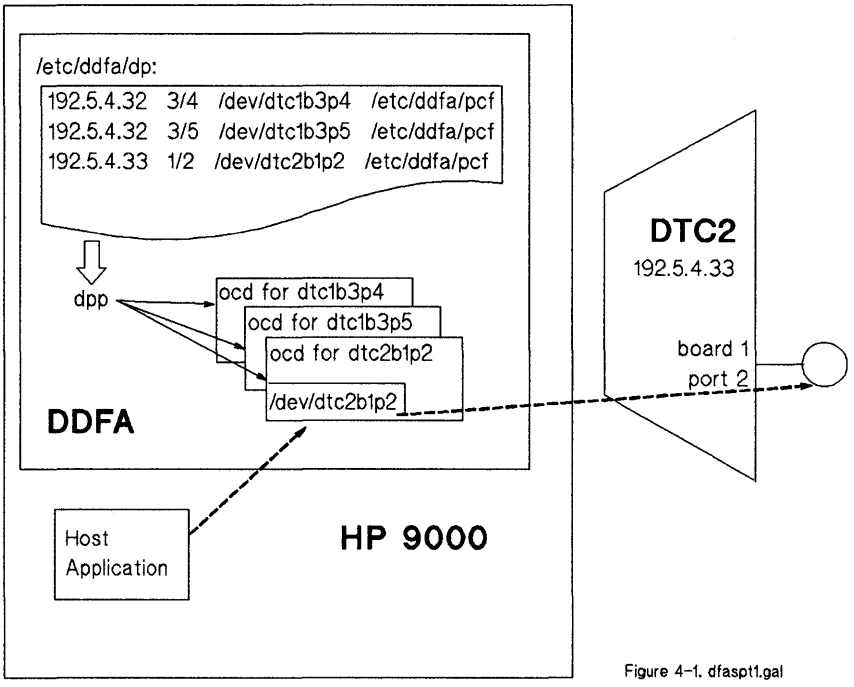


Figure 4-1. HP 9000 and DTC Interaction With DDFA

Incoming Connections Without Telnet Port Identification

An incoming connection is a connection *initiated by the DTC* to the host. For example, a user tries to login from a terminal on a DTC. The terminal is requesting a response from the system to login.

Every time a MUX-connected terminal logs onto a system, the device file associated with the session is the same. The user can find out what MUX port the terminal is connected to by using the HP-UX **tty** command. For example, the device file name shown below from executing the **tty** command is **/dev/tty2p3** for MUX card 2, port 3.

```
tty
/dev/tty2p3
```

The Telnet daemon (**telnetd**) assigns a pty to the connection when a user logs into the system from a terminal on a DTC. The pty device files refer to logical devices, and the Telnet daemon selects them randomly from the pool of free ptys in the **/dev** directory. Even though you can use the HP-UX **who** or **tty** command to find the name of the device file associated with your Telnet session, the result does not show which DTC port is yours. The assigned pty is different each time you login even from the same terminal.

Figure 4-2 illustrates the system and DTC interaction without Telnet port identification. A user at a terminal tries to login, and the connection is assigned a device file from the pty pool.

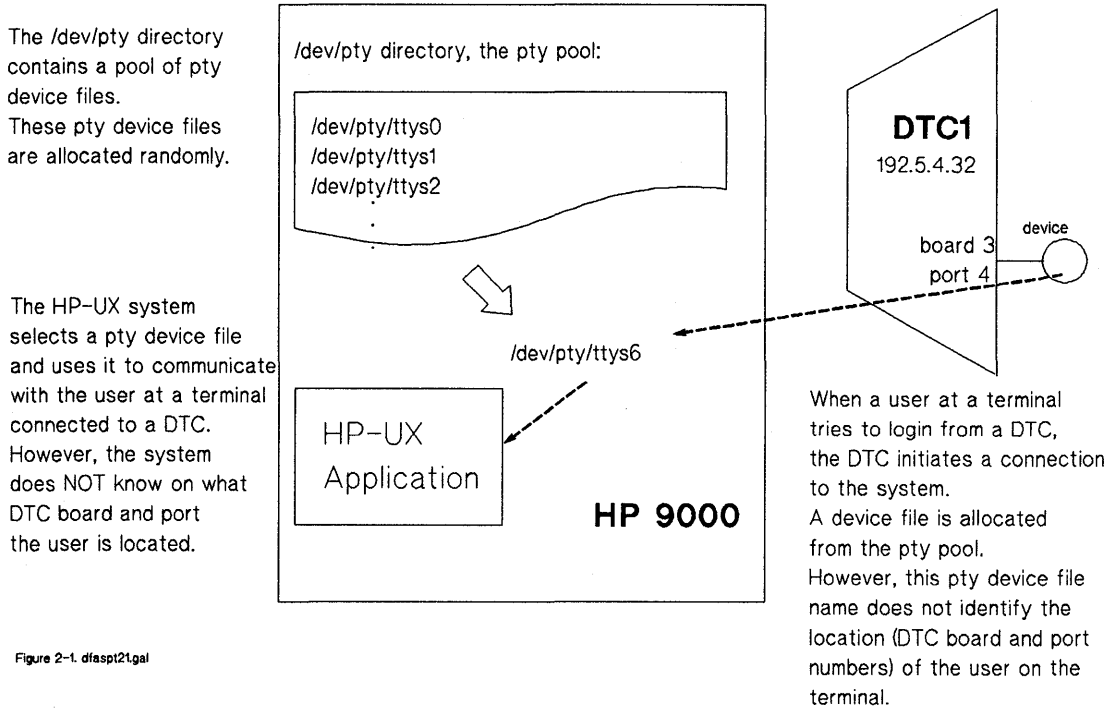


Figure 4-2. Devices Assigned From the Pty Pool Without Telnet Port Identification

How Telnet Port Identification Helps

Telnet Port Identification refers to the ability of the Telnet daemon to assign incoming Telnet connections to specific pty device files, based on the origin of the Telnet connection. When the system accepts an incoming Telnet connection, it asks the calling DTC to give it the board and port numbers of the DTC port. If a mapping between the DTC port and a pty device file has been defined in the **/etc/ddfa/dp file**, then the defined pty device file is used to service the incoming connection. In essence, the randomness in pty assignments for incoming connections has been eliminated.

If the mapping between the DTC port and a pty device file has not been defined in the **dp** file, then the pty is assigned randomly. The Telnet daemon choses a pty device file randomly from the pool of free ptys in the **/dev** directory. This pty device file is then associated witht the connection between the DTC port and the system.

Figure 4-3 illustrates the system and DTC interaction with Telnet port identification. Note that entries for incoming connections in the **dp** file do not have **pcf** files associated with them.

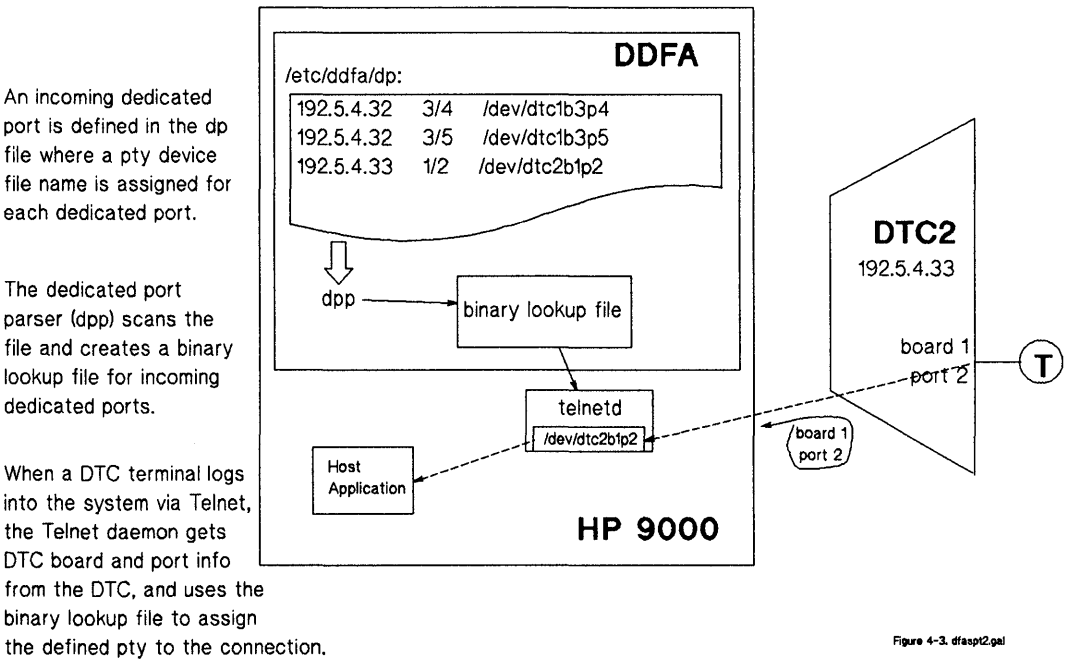


Figure 4-3. Incoming Connection Using Telnet Port Identification

Examples of Differences Between tty and pty Devices

The table below illustrates some ways that the DDFA Utilities and Telnet port identification change the way that pty devices are accessed.

Table 4-1. Differences Between Accessing tty and pty Devices

TTY Devices	PTY Devices	PTY Devices when using DDFA Utilities and/or Telnet Port Identification
The tty device files refer to specific, physical devices.	Pty device files do not refer to specific physical devices. They are assigned randomly from a pool of free ptys in **/dev** at connection establishment time.	Well-known pty device files can be associated with specific physical devices connected to DTC ports.
Applications can refer to tty devices (e.g. printers) by using the device file name.	An application cannot be configured with a pty device file because the device file is selected randomly, and not until the connection has been established.	Well-known pty device files can be defined and referred to at application configuration time. In addition, DDFA establishes the connection to the DTC port when the pty is opened by the application.
The HP-UX spooler can be configured to use a tty printer as an output device.	Since pty device files do not specify a unique device, there is no way to configure pty printers with the HP-UX spooler.	Pty device files can be predefined and then used to configure DTC printers as HP-UX Spooler output devices.

Table 4-1. Differences Between Accessing tty and pty Devices (continued)

TTY Devices	PTY Devices	PTY Devices when using DDFA Utilities and/or Telnet Port Identification
The **getty** processes are spawned on specific physical ports. Therefore login sessions are "dedicated" to that port.	The **getty** processes are spawned by **telnetd** on the randomly selected pseudoterminal. Therefore, the physical origin of a login session cannot be determined.	Based on the calling DTC port, the Telnet daemon initiates a login process on a predefined pty name.
The **tty** command and the **ttyname**() system call return a file name which refers to a specific physical port.	The **tty** command and the **ttyname**() system call return the name of the randomly-selected pty device file.	The **tty** command and the **ttyname**() system call return the name of a well-known pty device file that refers to a specific DTC port.

5

DDFA Configuration

This chapter explains **dp** (dedicated port file), **pcf** (port configuration file), and **dpp** (dedicated port parser) in more detail.

The **dp** file defines the association of physical DTC ports to logical HP-UX pty device files. A DTC port associated in this manner is referred to as a dedicated port, because of its fixed correspondence to a specific device file.

Incoming and Outgoing Dedicated Ports

Dedicated ports can be defined as either incoming or outgoing ports.

Incoming ports typically have user terminals on the DTC port. The system accepts incoming login sessions from incoming dedicated ports. The important point is that the DTC initiates the incoming connections from incoming dedicated ports.

Outgoing dedicated ports typically have output devices such as printer, plotter, or terminals used as output-only devices attached to the DTC port. However, an outgoing dedicated port may also have an input/output device on the DTC. The important point here is that the system, not the DTC, initiates and opens outgoing connections to outgoing dedicated ports.

Using Dedicated Ports

Use of dedicated ports on HP-UX may involve both the DDFA Utilities for outgoing connections and the Telnet Port Identification feature for incoming connections (but not simultaneously on the same device file) . Only system administrators concerned with DTC device configuration need be concerned with these features. Once the device files are defined via DDFA, everyone can use them as they normally do. In addition, assuming the ARPA Telnet service is configured properly and running on the host, the administrator who configures the dedicated ports need not be concerned with ARPA Telnet operation or configuration.

Configuration of the DDFA Utilities and Telnet Port Identification features centers around the Dedicated Port File, **/etc/ddfa/dp**. The system administrator defines the mapping of DTC ports or devices with pty names in this dedicated port file. Later, application configurators or system programmers will use these pty names to refer to DTC devices.

The Dedicated Port File, dp

Dedicated ports are configured in the dedicated port file, **/etc/ddfa/dp**. The **dp** file is an ASCII file which consists of entries, one for each dedicated port, of the following format:

For outgoing connections (ports for output devices such as printers and plotters)

```
<DTC IP address> <board>/<port> <device file name> <port config
file>
```

For incoming connections (ports for input devices such as terminals)

```
<DTC IP address> <board>/<port> <device file name>
```

where the fields are defined as:

`DTC IP address` — The IP address assigned to the DTC or assigned to the port on the DTC. The IP address is configured on the DTC and DTC ports by using the DTC Manager. Refer to the product documentation for the DTC Manager/UX product (J2120A) or the PC-based DTC Manager product (D2355A).

`board` — The physical DTC board of the port to be dedicated. Refer to the DTC hardware documentation for number of boards per each DTC type.

If `xx/xx` is specified for `<board>/<port>`, then the specified IP address references a port on the DTC and not the DTC CPU. This `xx/xx` parameter can be used to form

	a pool of ports. Refer to "Pooling Devices" in Chapter 6, "Application Examples."
/	A slash separates the board and port numbers.
port	The port number of the DTC port to be dedicated. Refer to the DTC hardware documentation for number of ports per board per each DTC type.
device file name	The name of a device file that will be used to access the dedicated DTC port. The device file name can be from 1 to 14 characters. The name of this file is created by the system administrator, and must not be the name of a file which already exists, or is already in use by the system. This file is created and managed solely by the DDFA Utilities and/or the Telnet daemon. HP recommends using a name which helps identify the DTC port being referenced, such as **/dev/dtc1b2p4** for DTC1, board 2, port 4.
port config file	For outgoing connections only. This file contains port configuration information for the DTC port. This information is used for the Telnet connection to the DTC. This file is **/etc/ddfa/pcf**. More information about this file can be found in the next section, "The Port Configuration File, pcf."

Refer to the figures in Chapter 4 for examples of how dedicated port files are used by DDFA.

The Port Configuration File, pcf

The port configuration file (**/etc/ddfa/pcf**) contains timer, connection, and data information for the related output device. The outgoing connection daemon, **ocd**, uses this information to manage the Telnet connection from the HP-UX device file to the DTC port.

The **pcf** file is required only for outgoing connections (connections initiated from the host to the DTC).

Each entry in the **dp** file which defines an outgoing dedicated port must refer to a port configuration file. In most cases, it is appropriate to use the default port configuration file, **/etc/ddfa/pcf**. You can use the same port configuration file for more than one dedicated port. Therefore, most entries in the dedicated port file will contain **/etc/ddfa/pcf** as the last field.

Whenever a **pcf** parameter or entry is changed, the **ocd** process that requires the changed parameter must be restarted. The **ocd** process must be restarted in order for those changed parameters to be effective.

Each entry in the **pcf** file is a field-value pair on a separate line. Each field-value pair is separated by a colon (:) or one or more spaces.

The master **pcf** file is in **/etc/newconfig/ddfa/pcf**. If different values are needed in a **pcf** file, the master file should be copied to another directory, such as **/etc/ddfa/pcf**. The copy should be modified and referenced in the **dp** file. HP recommends that you create the **/etc/ddfa** directory to contain the **pcf** and **dp** files.

The **pcf** file contains the following entries and defaults:

`telnet_mode: enable`
: Perform data transfer using the Telnet protocol. This option *must* be enabled for using the DTC. If disabled, no connection to a DTC can occur.

`timing_mark: enable`
: Use the Telnet timing mark negotiation at the end of the data transfer. All data is output from the DTC buffers to the device before the buffers are flushed.

`telnet_timer: 120`
: Set the timeout for the timing mark negotiation to 120 seconds. If the negotiation does not complete within 120 seconds, an error message is logged to **/usr/adm/syslog**, and the error is sent to the user application program.

`binary_mode: disable`
: Transfer data in ASCII mode. Do not ignore processing of special characters such as XON/XOFF. If set to "enable," the data transfer over the network will be in binary mode and treatment of special characters (for example XON and XOFF) will not occur.

  Because there is no flow control, data integrity cannot be guaranteed when binary mode is enabled.

	If binary mode is disabled, it may still be negotiated by the application program setting IXON to 0 (zero) in the TERMIO data structure.
`open_tries: 1500`	Set the number of connection retries to 1500. If the retry process fails to make a connection, an error message is logged to **/usr/adm/syslog**. The error message is also transmitted to the user application program. The retry process can be interrupted by sending the SIGUSR2 signal to the **ocd** process using the **kill -17** command.
`open_timer: 30`	Set the time between retries for making a connection to 30 seconds.
`close_timer: 0`	Set the time between an application program's **close** call and the actual close of the connection to zero seconds. The connection will be closed and opened after every file is sent to the device. Setting the timer to a value other than zero avoids the overhead of opening and closing the connection when a spooler spools several files at a time. The connection closes after the specified length of time. Setting the timer to a high value effectively leaves the connection permanently open.
`status_request: disable`	Disable the sending of a status request to a DTC printer. If enabled, a status request is sent to the device. The device replies with a status such as busy or ready.
`status_timer: 30`	Set the timeout for receipt of a status reply to 30 seconds. If no status reply is received, an error message is logged to **/usr/adm/syslog**. An error is also sent to the user application program.
`eightbit: disable`	Cause the eighth data bit (bit 7) to be stripped by the pty. If enabled, the eighth data bit is not stripped.
`tcp_nodelay: enable`	Cause data to be sent to the LAN as soon as it is received by TCP.

Managing Outgoing Dedicated Ports With dpp and ocd

Each outgoing dedicated port is managed by an outgoing connection daemon, **ocd.**

Once all outgoing dedicated ports have been defined in the dedicated port file, **/etc/ddfa/dp**, the DDFA Utilities can be started with the **dpp** command (see next paragraph). The result of this step will be that one outgoing connection daemon (**ocd**) process will be running for each valid entry defined in the dedicated port file. The dedicated port parser, **dpp**, scans the dedicated port file and spawns an **ocd** process for each entry in the file.

The **dpp** program can be run manually, or included in a system startup file such as **/etc/netlinkrc** to be executed automatically at system start up time.

The syntax for **dpp** is:

```
dpp <dp_file> [-l <logfile>] [-c] [-k] [-p<ocd_program>]
```

`dp_file`	Name of the dedicated port configuration file to use. Typically, the **dp** file is **/etc/ddfa/dp.**
`-l logfile`	Name of the log file to which **dpp** should log errors. If not specified, the error messages are logged to the system console screen.
`-c`	Parses the dp file and logs all bad entries without executing **dpp** fully. It is useful to debug the **dp** file before running it. The **-p** option is ignored if **-c** is used.
`-k`	Removes each device file from the **/dev** directory which is also in the **dp** file. Then, **dpp** spawns an **ocd** daemon for each valid entry in the **dp** file. Refer also to "Using dpp" later in this chapter. The **ocd** daemon normally creates and removes the pty device files associated with DTC ports. However, if the **ocd** process is killed improperly, a device file may remain. If the system is rebooted, the **-k** option may be specified to delete

any remaining device files and to restart all the **dp** file entries correctly.

`-p<ocd program>` The default path for **ocd** is **/etc/ocd**. If the path is different, it must be specified with the **-p** option. The **ocd** must be executed with access rights for **dpp**.

Full pathnames must be specified for the **dp** file and log files.

The **dpp** command performs the following actions for *each* outgoing dedicated port defined in the **dp** file:

- Checks the validity of the device file name.
- Runs an **ocd** process on the pty which was created. The **ocd** process manages the outgoing Telnet connection to the DTC port, establishing it when the pty file is opened.

  In addition, **ocd** maps the predefined device files to the ptys in the pty pool (in **/dev/pty** and **/dev/ptym**). Therefore, it is important to insure that the number of ptys defined in the system pool in the **uxgen** file is sufficient. Consult the **insf** command man page for more information about creating ptys for the system pool.

After running **dpp**, you can check to see that the **ocd** processes are running by using the HP-UX **ps -ef** command from the Shell prompt as follows:

```
ps -ef | grep ocd
```

After the **ocd** processes have been started, accessing the pty device file (opening the slave side of the pty) results in access to the associated DTC device.

Using dpp

Dpp cannot execute an **ocd** process for an entry in the **dp** file if the pty device file specified in that entry already exists. The existence of such a file could indicate that an **ocd** process is already running for that entry, or that another application is already using the device file. Therefore, it is important to kill all running **ocd** daemons properly and remove the device files they use before using **dpp** to start or restart the **ocd** daemons.

Stopping the DDFA Utilities involves two steps.

- First, the **ocd** processes must be killed.
- Second, the device files that the **ocd** processes use must be deleted from the file system.

Sending a signal 15 to an **ocd** process causes both of these things to happen automatically. For example,

```
kill -15 <pid of ocd>
```

The pid is the process identification number of the **ocd** daemon.

The **kill -9** command also kills the **ocd** process, but does not remove the device file. If the **kill -9** command is used, you will have to remove the device file manually from the **/dev** directory using the HP-UX **rm** command. Therefore, the **kill -15** command is more complete.

Using dpp with the -k option

Running the **dpp** program with the **-k** option causes **dpp** to kill all **ocd** processes for ports defined in the **dp** file and remove their associated pty device files. In addition, it restarts **ocd** processes.

Using **dpp -k** at system startup time (in the **/etc/netlinkrc** script) is an excellent way to start up the **ocd** processes. Then all **ocd** processes defined in the **dp** file are started properly and old files removed even if the **ocd** processes were previously aborted.

The syntax is:

```
/etc/dpp   /etc/ddfa/dp   -k
```

An example of adding **ocd** sections to the **/etc/netlinkrc** script is as follows:

```
ocd_start()
{
  /etc/dpp /etc/ddfa/dp -k
}
```

Execute the **ocd** startup subroutine after the network services:

```
case standalone )
  ...
  net-start
  ocd_start     # add this line
  ...
;;
```

Using dpp After Adding Outgoing Entries to the dp File

After adding entries for outgoing dedicated ports to the **dp** file, run **dpp** without the **-k** option. The **-k** option causes **ocd** processes for existing entries to be killed. Aborting the existing **ocd** processes disrupts service to any current active sessions. To start **ocd** processes for the newly added entries to the **dp** file, execute **dpp** *without* the **-k** option. The syntax is:

```
/etc/dpp   /etc/ddfa/dp
```

Ocd processes will be started only for the newly added entries. Dpp will ignore entries that already exist.

Using dpp After Removing Outgoing Entries From the dp file

If you remove outgoing entries from the **dp** file, be sure to kill their associated ocd processes using the **kill -15** command. This command will insure that the **ocd** processes and their associated device files are removed properly. The syntax is:

```
kill -15 <pid of ocd>
```

The <pid of ocd> is the process identification number of the **ocd** daemon.

Managing Incoming Connections by telnetd

Incoming connections are initiated by the DTC (and not by the system) and are Telnet connections. Incoming Telnet connections are handled by the Telnet daemon (**telnetd**). The Telnet daemon uses the Port Identification feature to assign a pty name for incoming DTC connections based on entries in the **dp** file.

If an incoming dedicated port is defined in the **dp** file, **telnetd** always uses the pty specified there. If the pty defined there is already in use, **telnetd** refuses the connection.

If an incoming Telnet connection is not from a DTC, or if the DTC Port is not defined in the **dp** file, then **telnetd** assigns a pty to the connection in the traditional manner, which is randomly from a pool of ptys.

Using dpp After Adding Incoming Entries to the dp file

The **dpp** process creates a binary file which encodes the information for the incoming dedicated port mappings defined in the **dp** file. Therefore, it is important to run **dpp** after making changes to the **dp** file. The syntax is:

```
/etc/dpp   /etc/ddfa/dp
```

Incoming connections on dedicated ports do not rely on the use of **ocd**. However, **telnetd** *does* require **dpp**, because **dpp** creates the binary lookup file that **telnetd** uses.

6

Application Examples

This section provides the following configuration examples for common uses of the DDFA Utilities and Telnet Port identification.

- Setting up printers with the HP-UX Spooler
- Accessing DTC ports programmatically
- Pooling devices

Setting Up Printers with the HP-UX Spooler

One very common use of DDFA Utilities is sending HP-UX spooler output to DTC printers. Refer to the *HP-UX System Administration Tasks Manual* for printer spooler information. Recall that when the HP-UX spooler is used with MUX-based printers, the printer is identified by its device file, such as, **/dev/lp**. Each printer also requires the configuration of a model script, which defines the manner in which the spooler communicates with a specific printer type. The system administrator configures this information using SAM, or using the following HP-UX commands from the Shell prompt: **lpadmin**, **enable**, and **disable**.

The HP-UX spooler can be configured to use DTC-based printers in exactly the same way as MUX-based printers are. The following tasks must be completed *before* configuring the HP-UX spooler:

1. The dedicated port file (**/etc/ddfa/dp**) must contain a dedicated port entry for the DTC printer.

2. An **ocd** process must be running for that port.

When configuring the HP-UX spooler, the pty device file name defined in the dedicated port file (**/etc/ddfa/dp**) identifies the printer. As usual, the system administrator selects a standard system model script that matches the printer connected to the DTC.

HP-UX Spooler Example

The Task: Configure the HP-UX spooler so that the default system printer in a DTC printer. Test the configuration by printing a file to the DTC printer using the HP-UX lp command. Use either SAM or HP-UX commands to configure the HP-UX spooler.

To configure the HP-UX spooler from SAM, follow these steps:

1. Select the following menu options:

   Peripheral Devices, Printers and Plotters

   Add a Local Printer

   Add a Printer with a Nonstandard Device File

2. SAM scans the system hardware and lists all the MUX cards on the system. Select any MUX card and any port on that MUX card. You will later change the MUX device file name to another device file name associated with the DTC.

3. Complete the **Add Printer** screen as follows:

Printer name:	Assign a name to the printer.
Printer model/interface:	Select an appropriate model script from **/usr/spool/lp/model**. Press **HELP (F1)** for a list.
Printer device file name:	Change the MUX device file name that SAM supplied to a device file from the **/etc/ddfa/dp** file for the DTC printer port.

4. Complete the other fields as desired.

5. Exit from SAM.

Another way to configure the printer is the command line method with the following commands:

```
# lpadmin  -pdtcprinter  -v/dev/dtc1b1p2   -mhp2235a
# accept dtcprinter
destination "dtcprinter" now accepting requests
printer "dtcprinter" now enabled
```

If desired, you can make dtcprinter to the the system default printer:

```
# lpadmin  -ddtcprinter
```

Now use the **lp** command to verify that your printer configuration worked:

```
# lp /etc/ddfa/dp
```

Accessing DTC Ports Programmatically

Programs that communicate with devices on a MUX, use the tty names defined in the **/dev** directory. Programmatic access to DTC ports relies upon using the outgoing dedicated ports. The outgoing dedicated port must be defined in the **dp** file, and an **ocd** process is running for that dedicated port. Then, the standard HP-UX **read**(), **write**(), **open**(), **close**(), and **ioctl**() calls can be used.

When using the **open**() call, the name of the device file for the path parameter should be a pty name defined in the **dp** file. When the **open**() executes, the outbound connection daemon, **ocd**, initiates a connection to the DTC port specified in the **dp** file. Then, read and write calls to the device can begin.

Use of the **ioctl**() call and the TERMIO structure is limited to actions which are supported by the DTC hardware and which are allowed over a networked connection. The main TERMIO restrictions include modem signal control and parity checking.

Refer to the **ddfa** manual reference page for more details on **ioctl** and TERMIO limitations. Refer to the **termio** manual reference page for details on terminal input/out, ioctl, and TERMIO structures.

Pooling Devices

Often it is necessary or desirable to pool several devices of the same class on one or more DTCs. For example, six modems could be allocated to a pool by assigning the same IP address to the six ports. The IP address must be different than the DTC CPU IP address. The **dp** file of DDFA can be configured to refer only to the IP Address, and not a board and port. The DTC then assigns connections to one of the six modem ports that is available.

IP addresses for each port and for the CPU on the DTC are configured on the DTC by using the DTC manager (either DTC Manager/UX, product J2120A, or the PC-based DTC manager, product D2355A).

Note All ports in a pool of devices must be on the same DTC. You cannot use the same IP address on more than one DTC. Furthermore, the network portion of a port IP address must be the same as the IP address for the DTC CPU.

Pooling devices is advantageous because it increases the chances that an attempt to access a device of a particular class is successful. Pooling allows a large number of users or applications to access several devices of the same type on a first-come, first-served basis.

The following example illustrates the pooling of six modems on a DTC. Each of two HP 9000s have multiple modem device files defined. System A has three modem device files defined, and system B has four modem device files. Each of the six ports has been assigned the same IP address from the DTC Manager.

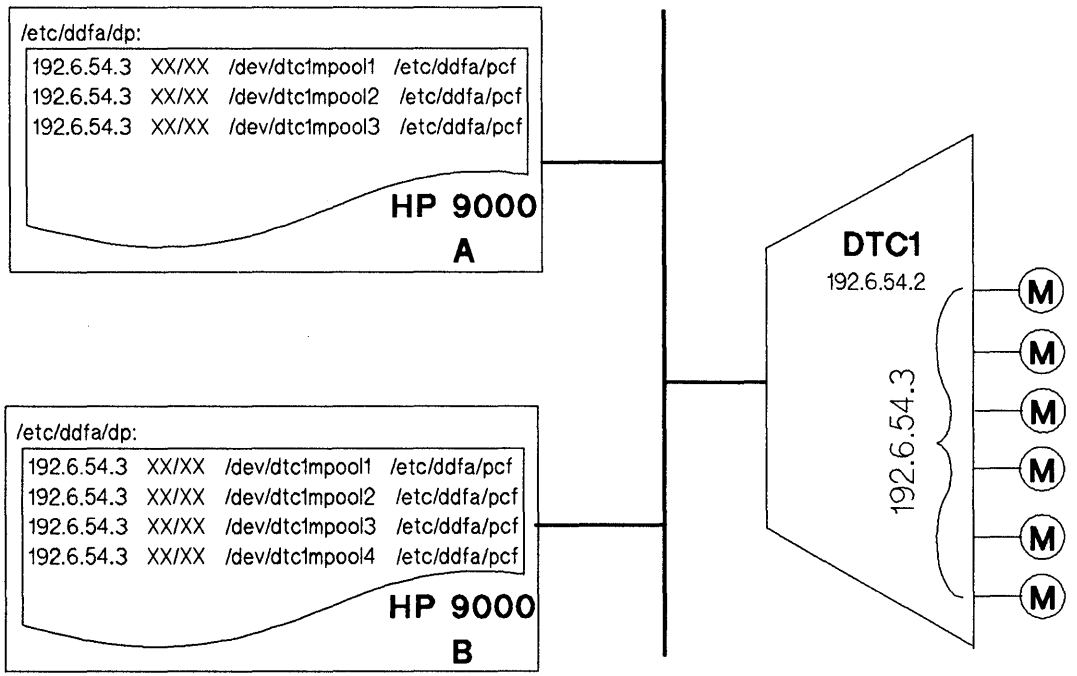


Figure 6-2. A Pool of Modem Ports on a DTC.

The steps involved are:

1. Using the DTC manager, configure each DTC modem port as required for the modem.

2. Choose a DTC port IP Address to reference the modem pool. This IP Address must have the same network portion as the DTC IP Address.

3. Assign this port IP Address to each modem port in the pool, by entering it in the Port Name field on the Port Configuration Screen of the DTC Manager.

4. On each host that will access the modems, define one or more dedicated ports in the **dp** file. However, instead of using `<board>/<port>` in the **dp** entry, use `XX/XX` or `xx/xx`. The entry `XX/XX` means that the specified IP Address references a destination port (or pool of ports), not a DTC.

When an application opens one of the device files, a Telnet connection is established. The DTC then assigns the first available modem port to the connection. Notice that in the above example there are seven device files defined and only six modems. Although only six users can access the modems *simultaneously*, more than six applications or users may contend for the six modems.

Another example for you to try is to configure a pool of printer ports on a DTC. Use two printers. Configure the DTC printers to operate with the spoolers on two separate HP 9000 systems. Print to one printer. While it is busy, start a print job from the other system. Verify that the DTC serves the second print job on the second printer in the pool.

Simple Troubleshooting

When troubleshooting dedicated connections involving DTC devices, it is important to make a distinction between *incoming* and *outgoing* connections, because these connections are serviced in different ways. Table 7-1 summarizes the difference between incoming and outgoing connections.

Table 7-1. Differences Between Incoming and Outgoing DTC Connections

Incoming Connections Originated by a DTC	Outgoing Connections Originated by an HP 9000 system
DTC board and port information is mapped to the system using the Telnet port identification feature.	The **ocd** daemon of the DDFA Utilities manages Telnet connections to DTC ports.
The **telnetd** daemon uses information in the dp file to assign well-known ptys to incoming connections.	The **dpp** spawns **ocd** processes for each outgoing connection specified in the **dp** file. The **ocd** daemon establishes an outgoing Telnet connection when the pty is opened.

Troubleshooting Outgoing Connections With ocdebug

Outgoing connections make use of the DDFA Utilities, **ocd** and **dpp**. Because **dpp** parses the **dp** file and tries to execute an **ocd** process for each valid entry, problems associated with **dpp** usually involve incorrect command line syntax, or references to illegal or non-existant files on the **dpp** command line. This class of problem will usually affect *all* ports configured in the **dp** file. The **dpp** error messages are listed in the error messages appendix of this manual.

If an outgoing connection on a *specific* DTC port does not function properly, the problem is likely to be with the **ocd** process. Usually, these problems will be a result of **ocd**'s inability to establish a connection to the DTC port given the parameters specified in the **dp** file or specified with the DTC Manager port configuration screen.

In the event of such a problem, the **ocdebug** program can be used. **Ocdebug** is identical to **ocd**, except that it provides logging messages on the user terminal or in a log file. The troubleshooter can start an **ocdebug** process on a terminal and examine the logged output to find problems.

The syntax for **ocdebug** is:

```
ocdebug -n<node name> -f<device_file> [-b<board>] [-p<port>]
[-c<port config file>] [-d<log_level>]
```

The syntax for **ocdebug** is the same as **ocd** except for the **-d** option in **ocdebug.**

The **-d** option indicates the level of debugging as follows:

0 No debug messages.

1 Log the trace procedure entry or exit.

2 Log additional tracking messages.

4 Dump data structures to **/usr/adm/ocd<pid>.**

The levels may be added together to accumulate debug capabilities. For example, **-d7** enables all levels, and **-d3** enables only the first two levels (log trace procedure and log tracking messages).

Debug messages are logged to the file **/usr/adm/ocd <pid>**, and the file name is displayed on your terminal at the start of debugging.

Here is an example of using **ocdebug**. To troubleshoot a connection to port 4 of board 3 on a DTC whose IP address is 15.13.115.123, *first* kill any existing **ocd** process running on that port, and then run **ocdebug**. See the example below.

```
kill -15 <pid of ocdebug>
ocdebug -n15.13.115.123 -focdebug -b3 -p4 -d4
```

This command would start an **ocdebug** process on the port and log to a file called **/usr/adm/ocd<pid>**. Contact your Hewlett-Packard representative to have this log file analyzed. **Ocdebug** also sends additional messages to **/usr/adm/syslog**. Output to **/usr/adm/syslog** is more readable and useful for basic troublshooting.

Troubleshooting Incoming Connections

If a DTC user logging into a system is assigned an unexpected device file, verify that the configuration of the user's port in the **dp** file is correct.

Since incoming connections do not use **ocd**, the **ocdebug** utility is of no use in troubleshooting problems associated with these connections. However, verification of the following points will usually solve any problems associated with incoming dedicated connections:

- Check that the proper configuration entry exists in the **dp** file. Remember that entries for incoming connections do not require the specification of a port configuration file (**pcf**).
- Make sure to execute the **dpp** utility after making changes to the **dp** file. Dpp creates a binary file which is used by **telnetd** to assign the proper device files to incoming connections.
- Check the version of the DTC Manager which manages the DTC. If the DTC Manager is PC-based, you should have version 12.1 or later. If it is the host-based DTC Manager, you should have version A.02.00 or later.
- Check the version of **telnetd** on the HP 9000 using the HP-UX **what** command.

  ```
  what /etc/telnetd
  ```

  On the $Header line display by the **what** command, the telnetd version should be 1.21.109.5 or later.

  ```
  $Header:  telnetd.c, v 1.21.109.5
  ```

- Make sure that the Telnet services are functioning properly.

Remember that the assignment of dedicated ptys on the HP 9000 depends upon the successful exchange of DTC board and port information by the DTC Telnet subsystem and the Telnet daemon on the HP 9000 at connection establishment time. Only the specified versions of software or later versions are capable of exchanging this information.

Migrating from Device Access/ARPA to DDFA

This appendix explains how to migrate from a similar product, Device Access/ARPA, to DDFA.

The Device Access/ARPA Product

Before the *DTC Device File Access Utiltities* were available, the *HP DTC Device Access/ARPA* product was used to access DTC devices. That product was included with the HP OpenView DTC Manager/UX (HP J2120A), or it could be ordered separately as an option to the PC-based DTC Manager (HP D2355A option AA0, AA1, or AAH).

The Device Access/ARPA software allows the HP-UX spooler to send output to DTC printers. In addition, the software provides a set of proprietary programmatic calls which can be used in user-written applications to send and receive data to specific DTC ports.

The DDFA Utilities functionally replace the older Device Access/ARPA software. Use of the DDFA Utilities is preferred over the use of Device Access/ARPA software because the method of access provided by the DDFA Utilities is more flexible and more transparent to the user.

The following table illustrates the functional equivalence of the DDFA Utilities and the Device Access/ARPA software.

Functional Equivalence of DDFA Utilities and Device Access/ARPA Software

Device Access/ARPA	DDFA Utilities
The HP-UX spooler is configured using DTC-specific model scripts. The configuration process is different from that of MUX printers.	After DDFA Utilities are installed and configured, the HP-UX spooler is configured using the standard HP-UX model scripts. Configuration of the spooler on the command line and within SAM is identical for DTC printers as it is for MUX printers.
User applications can communicate with DTC devices by using an HP-proprietary set of programmatic calls to open and close connections to DTC ports and read and write to them.	User applications can communicate with DTC devices by using the standard HP-UX read(), write(), open(), close(), and ioctl() calls.

Note Hewlett-Packard recommends that customers currently using the Device Access/ARPA software migrate to the DDFA Utilities.

An Example Migration

This section describes the steps necessary to migrate from a printer configured using DTC Device Access/ARPA to a DDFA-based configuration.

Assume that a printer called **dtcprinter** has been configured using Device Access/ARPA. The printer is connected on a DTC called **dtc2.grenoble.hp.com** on **port 0** of **board 1**. The printer is an **HP2335A** printer, and the model script is **/usr/spool/lp/pr_serv/model/pr_hp2335A.**

1. First, get all of the configuration parameters for that particular printer, **dtcprinter**.

   Using the **lpstat** command, get the configuration file used for that printer:

```
# lpstat -t
scheduler is running
system default destination: dtcprinter
```

```
device for dtcprinter: /usr/spool/lp/pr_serv/conf/dtcprinter
dtcprinter accepting requests since Dec 11 12:23
printer dtcprinter is idle. enabled since DTC 11 12:24
```

Verify that no printer requests are pending or waiting in the spooler queue. If so, you should wait, or write down the file name so that you can resubmit these print jobs after reconfiguring the spooler.

Note that the Device Access/ARPA configuration file for **dtcprinter** is reported by **lpstat** as the device for **dtcprinter**.

2. Read the configuration file for **dtcprinter**.

```
# cat /usr/spool/lp/pr_serv/conf/dtcprinter

pr_serv.destination_name        dtc2
pr_serv.service_name            dtc2blp0
pr_serv.add_cr_on_end_of_line   enable
pr_serv.status_checking         disable
pr_serv.status_timer            120
pr_serv.telnet_timer            120
pr_serv.log_on_console          enable
pr_serv.telnet_mode             enable
pr_serv.log_file                /usr/spool/lp/pr_serv/model.log
pr_serv.pr_serv.save_directory  /tmp/save
pr_serv.abort_spooler_on_error  disable
pr_serv.open_retries            1500
pr_serv.open_timer              30
```

3. Using the destination_name, find the IP Address of the DTC, **dtc2**:

```
# nslookup dtc2

Name Server:  hpgneds3.grenoble.hp.com
Address:  15.128.8.10

Name:  dtc2.grenoble.hp.com
Address:  15.128.9.222
```

4. Now configure DTC Device File Access Utilities (DDFA) for access to the HP2335A printer via a standard HP-UX device file.

To complete this step, you need the following four pieces of information:

DTC IP Address:	15.128.9.222
Physical Location:	Board 1, Port 0
Model Script:	**/usr/spool/lp/model/hp2235a**
Port Configuration File:	**/etc/pcf/newconfig/ddfa** or **/etc/ddfa/pcf**

Note that a standard HP-UX model script and the default Port Configuration File, **pcf**, supplied with the DDFA Utilities are used here.

Add an entry to the dedicated port configuration file, **/etc/ddfa/dp**, for the printer. The new entry should look like:

```
15.128.9.222   01/00   /dev/dtc1b1p0   /etc/ddfa/pcf
```

Enable these modifications by running the dedicated port parser, **dpp**:

```
# /etc/dpp /etc/ddfa/dp
```

5. Verify that the device file, **/dev/dtc1b1p0**, has been created by the **ocd** process which was spawned by **dpp**. The **ll** command displays this information:

```
# ll /dev/dtc1b1p0
crw-rw-rw- 1 root other 17 0c000038 Dec 11 19:25 /dev/dtc1b1p0
```

6. Before configuring the spooler, verify that you can access the printer via its device file, **/dev/dtcprinter**:

```
# cat /etc/ddfa/pcf  /dev/dtc1b1p0
```

Check that the **pcf** file was correctly printed on the HP2335A printer.

7. Verify again that no files are in the printer queue, and then stop the spooler and delete **dtcprinter** from the spooler configuration.

```
# lpstat -t
# lpshut
scheduler stopped
# lpadmin -xdtcprinter
```

8. Now configure the spooler using the new parameters for **dtcprinter** (see Step 3):

```
# lpadmin -pdtcprinter -v/dev/dtc1b1p0 -mhp2235a
# accept dtcprinter
destination dtcprinter now accepting requests
# enable dtcprinter
printer dtcprinter now enabled
```

And if desired, make **dtcprinter** the system default printer:

```
# lpadmin -ddtcprinter
```

9. Before starting the spooler again, verify that the printer is correctly configured using **lpstat**:

```
# lpstat -t
scheduler is not running
system default destination: dtcprinter
device for dtcprinter: /dev/dtc1b1p0
dtcprinter accepting requests since Dec 11 13:23
printer dtcprinter is idle.  enabled since Dec 11 13:24
```

10. Start the spooler:

```
# lpsched
```

11. Verify that the printer spooler is correctly configured to send output to **dtcprinter**:

```
# lp /etc/ddfa/pcf
request id is dtcprinter-1 (1 file)
```

12. Remove the Device Access/ARPA product files after migrating all DTC devices:

```
# cd /usr/spool/lp
# rm -r pr_serv
```

Caution The **rm -r** command shown above causes the pr_serv directory and all of its subdirectories to be deleted.

---

13. End of migration.

Appendix B: dpp Error Messages

Error Number	Message	Reason
0	<dp_file>is mandatory.	No **dp** file was found when **dpp** was started.
1	<dp_file> must be the first argument.	The **dp** file must be specified as the first argument when **dpp** is started.
2	Cannot read dp file.	The **dp** file mentioned does not exist or cannot be accessed with current access rights.
3	No log file defined (-l option).	The **-l** option has been detected with no arguments.
4	Cannot create (-l option).	The log file cannot be created because of invalid path or insufficient access rights.
5	Cannot access log file (-l option)	The log file cannot be accessed because of invalid path or insufficient access rights.
6	No **ocd** file defined in program option.	The option **-p** has been detected with no arguments.
7	Can't execute **ocd** program.	The **ocd** program does not exist or is not an executable file with correct access rights.
8	Cannot purge device file (/dev/*xxx*).	The option **-k** has been selected and the device file exists but cannot be purged because of insufficient access rights.

9	Cannot execute default program (/etc/ocd).	The **ocd** program cannot be executed because of insufficient access rights or because it has not been properly installed.
10	Entry ignored (IP address).	The mentioned entry of the **dp** file does not have a valid IP address.
11	Entry ignored (no port/board info).	The entry is ignored because **board/port** information was not mentioned in the **dp** file.
12	Entry ignored (bad port number).	The port mentioned in the displayed entry is not either a decimal value, or a string composed of **x** or **X** characters.
13	Entry ignored (bad board number).	The board mentioned in the displayed entry is not a decimal value, or it is not a string composed of **x** or **X** characters.
14	No more processes available on system.	The **ocd** program cannot be started because of lack of processes in the system.
15	Entry ignored (no device name).	The device file name has not been configured in the **dp** file.
16	Entry ignored (bad device name).	The displayed device file cannot be created becuase of invalid path or insufficient access rights.
17	Entry ignored (bad config name).	The displayed configuration file cannot be read because of invalid path or insufficient access rights.

Manual Pages

This section contains the HP-UX Online Manual Pages included with the DDFA software. Please consult the online manual pages for the most up-to-date information. In addition, be sure to read any README files that might be included with your system.

The following manual pages are provided here:

ddfa(7)	DTC Device File Access utilities description
dp(4)	Dedicated port file
dpp(1m)	Dedicated port file parser
ocd(1m)	Outbound connection daemon
ocdebug(1m)	Debug version of ocd
pcf(4)	Port configuration file

ddfa(7)

Name

ddfa - HP DTC Device File Access software

Description

The HP Datacommunications and Terminal Controller (DTC) Device File Access (DDFA) software allows access from HP-UX systems and user-written applications to HP DTCs using standard HP-UX structures. DDFA provides an interface to remote (LAN-connected) DTC ports which is similar to the interface for local mux ports.

The basic principle is that a daemon is created for each configured port based on information in a configuration file (dp file). When the daemon is spawned, it takes a pty from the pool and creates a device file with the same major and minor number as the pty slave. The device file is known as the "pseudonym", and user applications use the pseudonym to access the server port using standard HP-UX intrinsics (open, close, write etc). The daemon listens on the pty until an application does an open(pseudonym), then manages the connection to the server port. The end result is that the server port is addressed via a device file, with the mechanism that makes it happen transparent to the user. For example, the lpadmin command can be used to set up a connection to a printer on a remote server port by using the pseudonym. A second configuration file (pcf) contains information to profile the port's behaviour.

DDFA consists of the following items:

`dp`

Dedicated Port configuration file. This is a text file which contains the information DDFA needs to set up and manage a connection to a specified DTC port. It contains a one line entry for each configured DTC port, which specifies the board and port, the pseudonym, and the path to the Port Configuration File (pcf).

The dp file is parsed by the Dedicated Port Parser (dpp) which spawns an outgoing connection daemon (ocd) for each connection specified in the file.

dp is also used by the HP-UX telnet daemon (telnetd) to identify incoming connections. In this usage, the daemon

negotiates the telnet environment option, and the DTC returns the board and port numbers of the connecting device. telnetd then looks up the port and board numbers in the dp file and maps them to the pseudonym. See dp(4) for more information.

There are two ways to specify a port: either by explicitly giving its IP address, or by giving the IP address of the server and then specifying the board and port. Alternately, the server's IP address and the TCP service port address of the port can be given.

`pcf`

Port Configuration File. This file is used by DDFA to configure specific port parameters. "pcf" is the generic name of the template file, in practice it is renamed for each port, and the values are altered appropriately for the device attached to the port. The pcf is referenced by an entry in the Dedicated Ports file (dp). The Dedicated Port Parser (dpp) parses the dp file and calls the Outbound Connection Daemon (ocd) program to spawn a daemon for each valid line in the dp file: a valid line is one in which the fourth field is the name of a pcf.

`dpp`

Dedicated Port Parser. dpp parses the Dedicated Ports file (dp) and calls the Outbound Connection Daemon (ocd) to spawn a daemon for each valid entry in dp. It can be run from the shell or it can be included in netlinkrc to automatically run the DDFA software each time the system is booted.

dpp has one mandatory argument, the path to the dp file. The other arguments specify an optional log file, the path to ocd if it is not the default, and an option to kill existing processes when the dp file is parsed. dpp can also be run in check mode, when it simply parses the dp file and reports any errors.

`ocd`

Outbound Connection Daemon. ocd manages the connection and data transfer to the remote DTC port. Normally, it is spawned by the Dedicated Port Parser (dpp), but it can be run directly from the shell. If it is run from the

shell, the name or IP address of the server or port and the pseudonym (device file name) must be entered on the command line. The board/port number or TCP service port address must be entered if the IP address does not explicitly name a port.

ocd creates a device file called ym < > problem for the connection. If ocd has an error condition which causes it to exit, it will normally remove the device file it created.

If a device file is accidentally deleted, the corresponding ocd will continue running for at least 30 seconds before it detects the absence of the device file. The ocd will then write an error message to /usr/adm/syslog and exit.

`ocdebug` Outbound Connection Daemon debug mode. ocdebug is a special build of ocd which contains debug code. ocdebug has to be run from the shell with all the arguments that would normally come from the dp file entered on the command line: apart from -d, the arguments are the same as the ocd arguments. The -d option specifies the level of debugging messages.

Installation

DDFA is automatically installed when the HP-UX system and ARPA are installed.

The files, their default directories and the correct access permissions are shown below. All files should be owned by bin with group bin:

```
-r-xr--r-- /etc/dpp
-r-xr--r-- /etc/ocdebug
-r-xr--r-- /etc/ocd
-rw-r--r-- /etc/newconfig/ddfa/pcf
-rw-r--r-- /etc/newconfig/ddfa/dp
-r--r--r-- /usr/man/manlm.Z/dpp.lm
-r--r--r-- /usr/man/manlm.Z/ocd.lm
-r--r--r-- /usr/man/manlm.Z/ocdebug.lm
-r--r--r-- /usr/man/man4.Z/dp.4
-r--r--r-- /usr/man/man4.Z/pcf.4
-r--r--r-- /usr/man/man7.Z/ddfa.7
```

Configuration

There are two basic steps to configuring the DFA software: (1) entering information in the dp file, (2) entering information in the pcf files.

Configuring the dp file

The dp file contains one line for each connection that is to be established. A template dp file is provided (/etc/newconfig/ddfa/dp): this can be edited directly or a separate file created. We recommend that the file is copied and the copy edited. We suggest you create the directory /etc/ddfa to contain dp and the pcfs. Note the following points:

- The default file is /etc/newconfig/ddfa/dp: if another file name or path is used, this information must be passed as an argument to dpp (see dpp(1m) for details) whether it is run from the command line or from netlinkrc.
- The exact order of the fields in dp must be maintained:

```
IP_address board/port pseudonym pc_file_path
```

- Fields can be separated by spaces or tabs.
- Only one entry is allowed per line.
- Comments can be appended using the # character: everything after the # is ignored by the dpp parser for each line.
- There are three ways to specify the DTC port: by explicitly giving its IP address, by giving the IP address of the server and then specifying the port and board, or by giving the IP address of the server and the TCP service port address of the port.
- It is recommended that the pcf and device file (pseudonym) are named appropriately so that both can be easily identified (for example, when listing processes with ps -ef). For example, the pcf could be named pcf_lp1 and the device file (pseudonym) ocd_lp1, or the pcf could be named pcf_dtc1b2p3 and the device file could be named ocd_dtc1b2p3.

`IP_address` — This is the IP address of the DTC being accessed, or the IP address of the port on the DTC.

`board/port` — This field contains the DTC board and port numbers, separated by the / character. The board and port numbers are not checked by dpp, but are checked by ocd. Valid values are 0 to 7 for the board, and 0 to 31 for the port

(these restrictions don't apply if a TCP service port address is given instead).

If the IP_address field explicitly defines the DTC port, the value in the port/board field must be xx/xx (you can use X or x).

If the entry is of the form xx/n where n is a decimal number, then n is taken to be the TCP port address, and this value is used when the connection is established, otherwise, the destination is filled using the formula (256*(32*board + port+1) + 23).

`pseudonym` This is the absolute path and name of the device file known to the system and/or the end user application.

The device file name is limited to 14 characters. We recommend that the name reflects the connected device, and is similar to the pcf name, for example ocd_dtc1b1p1.

`pc_file_path` This is the path to the Port Configuration File (pcf) which contains the configuration information for the DTC port. This field is mandatory, as the parser, dpp, uses the presence of this field as its flag to spawn a daemon for the line. We recommend that the name of the file reflects the connected device, and is similar to the pcf name, for example pcf_dtc1b1p1.

The following examples illustrate the usage:

```
11.234.87.123 03/01 /dev/ocd_lp1 /etc/ddfa/pcf_lp1 # lp1 b1,n2,f7
```

This example refers to a printer connected to port 1 of board 3 of a DTC with the IP address 11.234.87.123. The device attached to the port can be accessed with the HP-UX spooler by using the device-file /dev/ocd_1p1 in the lpadmin command. The port will be profiled using data in the file /etc/ddfa/pcf_lp1.

```
11.234.87.124 xx/xx /dev/ocd_lp2 /etc/ddfa/pcf_lp2 # lp2 b2,n1,
```

The above example refers to a printer connected to the DTC port with the IP address 11.234.87.124, so the board/port field contains xx/xx. The file pcf_lp2 contains port profiling information.

```
11.234.87.215 xx/16919 /dev/ocd_lp3 /etc/ddfa/pcf_lp3 # lp3 b2,p1
```

The above example shows how a port can be specified using a TCP port address. The port address is calculated using the formula (256*(32*board + port+1) + 23).

Configuring the pcfs

Port Configuration Files (pcf) are used to configure individual server ports. /etc/newconfig/ddfa/pcf is a template file, in practice it should be renamed for each port, and the values it contains altered to suit the device attached to the port. We recommend you create the directory /etc/ddfa to contain the modified pcfs and the modified dp file. The pcf is referenced by an entry in the Dedicated Ports file (dp).

The Dedicated Port Parser (dpp) parses the dp file and calls the Outbound Connection Daemon (ocd) program to spawn a daemon for each valid line in the dp file: a valid line is one in which the fourth field is the name of a pcf.

Note the following points:

- The order of the variables is not important.
- The file consists of the names of variables and their values. The variables in the template file are shown terminated by a colon (:), but this is not mandatory.
- The variables and their values can be separated by spaces or tabs.
- Only one variable-value pair is allowed per line.
- Only the value should be altered, the variable name must remain the same.
- If you want to use the default values, then the template file /etc/newconfig/ddfa/pcf can be referenced in the dp file.
- If you want to use different values for a port, you should make a copy of /etc/newconfig/ddfa/pcf: it is a good idea to name the file appropriately for the pseudonym (device file), for example, pcf_dtc1b2p3.

The file contains the following information:

`telnet_mode:` This can have the value disable or enable.

When it is enabled, data transfer over the network will use the Telnet protocol. This option must be enabled for the DTC.

`timing_mark:` This can have the value enable or disable.

When it is enabled, a telnet timing-mark negotation will be sent to the DTC after all user data has been transferred. ocd will wait for a reply to the timing mark negotiation before closing the connection. This ensures that all data has been output from the DTC buffers to the device before the buffers are flushed. It should therefore be enabled for the DTC.

`telnet_timer:` This defines the time, in seconds, during which the software will wait for a response to the telnet timing mark and binary negotiation.

If the timer pops, an error message will be logged to /usr/adm/syslog and the error will be transmitted to the user application.

`binary_mode:` This can have the value disable or enable. When it is enabled, data transfer over the network will be in binary mode, and treatment of special characters (for example XON/XOFF) will be disabled.

Due to the absence of flow control, data integrity cannot be guaranteed when binary_mode is enabled.

Note that even if binary_mode is disabled, it may be negotiated at any time by the application setting IXON to 0 in the termio data structure.

`open_tries:` This defines the number of times the software will try to open a connection before giving up. If the value is 0 the software will try "forever" (approximately 68 years). If the retry process fails, an error message is logged to /usr/adm/syslog. The error message is also transmitted to the user application.

The retry process can be interrupted by sending the SIGUSR2 signal to the ocd process using kill -17 .

Note that if the application exits after asking ocd to open the connection to the DTC, ocd will continue trying to open until open_tries and/or open_timer are exceeded.

`open_timer:` This defines the time in second between tries. If the value is 0 ocd uses an exponential retry period algorithm up to 32 seconds, that is, 1 2 4 8 16 32 32 32 ...

`close_timer:` This defines the time in seconds between the close call made by the application on the pty slave and the moment when the connection is actually closed. Setting this value to, for example, 5 seconds avoids the overhead of opening and closing the connection when a spooler spools several files at a time. Setting a sufficiently high value effectively leaves the connection permanently open.

`status_request:` This may have the value disable or enable.

When it is enabled, the software sends a status request to the printer attached to the server and processes the reply as follows:

`LP_OK (0x30)`
ocd continues processing.

`LP_NO_PAPER (0x31)`
ocd retries within the limits of the status timer.

`LP_BUSY (0x32)`
ocd retries within the limits of the status timer.

`LP_OFF_LINE (0x23)`
ocd retries within the limits of the status timer.

`LP_DATA_ERROR (0x38)`
ocd retries within the limits of the status timer.

`status_timer:` This defines the time, in seconds, after which the software no longer waits for the reply to the status request. If the timer pops, an error message is logged to /usr/adm/syslog. The error condition is also transmitted to the user application.

`eightbit:` This can have the value enable or disable.

Normally, data bytes processed by the pty have bit 7 stripped.

If eightbit is enabled, the stripping is disabled. If eightbit is disabled, stripping is enabled, and bit 7 is stripped. This can also be achieved by changing the pseudonym's termio structure using the ioctl commands.

`tcp_nodelay:` This can have the value enable or disable If it is enabled, data is sent to the LAN as it is received. It can be disabled if the software is sending packets faster than the server can accept.

The default values are:

```
telnet_mode: enable
timing_mark: enable
telnet_timer: 120
binary_mode: disable
open_tries: 1500
open_timer: 30
close_timer: 0
status_request: disable
status_timer: 30
eightbit: disable
tcp_nodelay: enable
```

Configuring netlinkrc

DDFA can be run at boot time by including a reference to dpp in netlinkrc. dpp will be run, the dp file will be read, and the appropriate processes created. It is recommended that the -k option is used, for example:

```
/etc/dpp /etc/ddfa/dp -k
```

Error Handling

When ocd receives a serious error condition, for example the LAN goes down, ocd will transmit the error conditions to the application by closing the pty. Any open,

close or write to the pseudonym will return the error condition, 0 bytes read. If the pseudonym is the controlling terminal for the group to which the application belongs, sighup will be sent to all the processes in the group, including the application.

Killing Daemons

Outbound Connection Daemons (ocds) should be killed using kill -15. kill -9 is not recommended as the device file remains. ocd will verify the validity of an existing pseudonym before trying to use it. dpp and ocd use data stored in the file /etc/utmp.dfa to verify whether a process still owns a pseudonym before taking it over. If ocd finds an unowned pseudonym, it will use it.

Ioctl Limitations

Not all ioctl functionality is available, due to the lack of a protocol which allows the transmission of such commands over the LAN to the remote port.

TERMIO Attribute Limitations

The main restrictions include modem signal control and parity checking. The following list shows those not available:

```
IGNPAR
PARMRK
INPCK
IXANY
IXOFF
CBAUD
```

IOCTL Request Limitations

This section lists IOCTL request limitations.

`TERMIO structure: CSTOPB flag`

DTC only supports one stop bit.

`TERMIO structure: CSIZE`

DTC only supports 8 bits per character. Value can't be modified.

TERMIO structure: PARODD flag

DTC offers static configuration to handle even or odd parity. It also handles auto parity detection for even or odd parity.

TERMIO structure: PARENB flags

Enabling/disabling done via static configuration. No programmatic interface supplied.

TERMIO structure: INPCK flag

No way to separate input from output parity features.

TERMIO structure: IGNPAR flag

Can't be configured on DTC.

TERMIO structure: PARMRK

Bad characters are forwarded to the system without marking them with OFFH,OH.

TERMIO structure: CBAUD

Speed is part of static configuration.

TERMIO structure: IXOFF flag

Flow control is enabled if the DTC static configuration specifies an ASCII access mode. If binary is selected, no flow control is provided.

TERMIO structure: IXON flags

Pacing of output to a terminal via a programmatic interface is enabed when ASCII mode is selected in static port configuration. It is disabled when binary mode is selected.

TERMIO structure: IXANY flag

DTC doesn't offer ability to restart output on any character received if XOFF was previously received.

TERMIO structure: HUPCL flag

DDFA doesn't support the hanging up of modem signals on the last close of the device file. If the modem signals used on the DTC drop, the connection is closed.

TERMIOstructure: CLOCAL flag

Not supported.

TERMIO structure: c_flags : IENQACK

Not supported.

c_flags: OFILL OFDEL NLDLY CRDLY TABDLY BSDLY FFDLY

Not supported by TELNETD-DTC software.

TERMIO structure: BINARY mode flags

Part of static configuration done in DTC Manager by selecting binary mode. If switching is enabled, binary can be selected at user interface level. There is no way to automatically negociate binary mode when proper termio flags are reset when using TELNETD. Binary/ascii switching is possible with DDFA. The DTC cannot support large reads in pure binary mode, so transferred blocks of data should not be more than 256 bytes. If half-duplex with remote acknowledgement is implemented, binary applications can be supported.

IOCTL System Calls

IOCTL: TCSBRK request

The ability to send a break without waiting for previous data to be sent is not provided at system level in TELNETD or DDFA. Receiving a telnet break command in the DTC will allow it to generate a break on asynchronous ports.

IOCTL: TCFLSH request

The DTC output queue cannot be flushed.

IOCTL: hardware handshake request.

Not supported on DTC.

IOCTL: TCXONC request

Local handshake cannot be disabled on DTC.

IOCTL: MCGETA request

Not supported.

IOCTL: MCSETA MCSETAF MCSETAW requests

There is no way to separately set modem lines of a DTC port.

IOCTL: MCGETT request

Modem timers CD timer, connect timer and disconnect cannot be configured.

CCITT/simple and direct/call in/call out modes

DTC cannot handle simple mode as there is programmatic interface for modem signals. Call-in mode cannot be simulated if the port is opened, as modem signals (or the call) must be present within 2 minutes otherwise the connection is cleared.

IOCTL requests: DACIDY get device adapter info

No way to get device adapter information.

IOCTL requests: download ioctl DACRADDR, DACDLADDR, DACDLGO DACDLVER

No programmatic call to download the DTC.

IOCTL requests: DACHWSTATUS, DACSELFTEST, DACLOADED, DACISBROKE

No programmatic interface to get such info.

IOCTL requests: DACLOOPBACK DACSUBTEST

port test

Files

```
/etc/dpp
/etc/ocdebug
/etc/ocd
/etc/dpp_login.bin
/etc/utmp.dfa
/etc/newconfig/ddfa/pcf
/etc/newconfig/ddfa/dp
```

See Also

dp(4) dpp(1m) ioctl(2) ioctl(5) ocd(1m) ocdebug(1m) pcf(4)

dp(4)

Name

dp - Dedicated Ports file, used by DDFA software and by DTC Port ID Via Telnet

Description

This file has two uses:

- DTC Port ID via Telnet

  The dp file is used by the HP-UX telnet daemon (telnetd) to identify the calling port and board of a telnet connection from an HP Datacommunications and Terminal Controller (DTC)

  At connection time, the host negotiates the telnet environment option, and the DTC replies with the port and board number of the connecting device. telnetd maps the port and board numbers to the well-known name for the device, which has previously been configured in the dp file.

- DTC Device File Access (DDFA)

  The dp file is used by the HP Datacommunications and Terminal Controller (DTC) Device File Access (DDFA) software to allow terminal server ports to be programatically accessed from HP-UX applications in the same way as devices connected directly to the HP-UX system.The dp file contains a one line entry for each configured DTC port.

The dp file contains the information that the DDFA software needs to set up and manage a connection to a specified DTC port. The file is parsed by the Dedicated Port Parser (dpp) which spawns an outgoing connection daemon for each connection specified in the file.

Port ID via Telnet

To configure the dp file for use in port id via telnet, the default file, /etc/newconfig/ddfa/dp, should be copied to a new file. The copied file should be configured with the appropriate values for the incoming connections. Create the directory /etc/ddfa to hold the dp file and the modified pcfs.

The information is in the following format:

```
dtc_ip_address board/port pseudonym
```

The exact details of each field are given in the CONFIGURATION INFORMATION section of this document.

DTC Outbound Connections

For DTC outbound connections, the following information is required:

```
dtc_ip_address board/port pseudonym config_file
```

The exact details of each field are given in the CONFIGURATION INFORMATION section of this document.

Configuration Information

There are three ways to specify the DTC port: by explicitly giving its IP address, by giving the IP address of the DTC and then specifying the port and board, or by giving the IP address of the DTC and the TCP port service address of the port.

Comments can be appended using the # character. Everything after the # is ignored by the dpp parser. The fields are separated by spaces.

Refer to ddfa(7) for information on how to configure and install the DDFA software.

The file has the following format:

`IP_address` — This is the IP address of the DTC being accessed, or the IP address of the port on the DTC.

`board/port` — This field contains the DTC board and port numbers, separated by the / character. It is not necessary to pack the values with leading zeros.

The board and port numbers are not checked by dpp, but are checked by ocd. Valid values are 0 to 7 for the board, and 0 to 31 for the port (these restrictions don't apply if the TCP service port address is given instead).

If the IP_address field explicitly defines the DTC port, the value in the port/board field must be xx/xx (you can use X or x).

If the entry is of the form xx/n where n is a decimal number, then n is taken to be the TCP port address. This value is used when the connection is established, otherwise, the destination is filled using the DTC formula (256*(32*board + port+1) + 23).

`pseudonym` This is the absolute path and name of the device file known to the system and/or the end user application. The device file name is limited to 14 characters. The name should reflect the connected device, and is similar to the pcf name, for example ocd_dtc1b1p1.

`pc_file_path` This is the path to the Port Configuration file (pcf) which contains the configuration information for the DTC port. This field is mandatory. The parser, dpp, uses the presence of this field as its flag to spawn a daemon for the line. The name of the file should reflect the connected device, and is similar to the pcf name, for example pcf_dtc1b1p1.

The following examples illustrate the usage:

```
11.234.87.123 03/01 /dev/ocd_lp1 /etc/ddfa/pcf_lp1 # lp1 b1,n2,f7
```

This example refers to a printer connected to port 1 of board 3 of a DTC with the IP address 11.234.87.123. The device attached to the port can be accessed with the HP-UX spooler by using the device-file /dev/ocd_1p1 in the lpadmin command. The port will be profiled using data in the file /etc/ddfa/pcf_lp1.

```
11.234.87.124 xx/xx /dev/ocd_lp2 /etc/ddfa/pcf_lp2 # lp2 b2,n1,
```

The above example refers to a printer connected to the DTC port with the IP address 11.234.87.124, so the board/port field contains xx/xx. The file pcf_lp2 contains port profiling information.

```
11.234.87.215 xx/16919 /dev/ocd_lp3 /etc/ddfa/pcf_lp3 # lp3 b2,p1
```

The above example shows how a port can be specified using a TCP port address. The port address is calculated using the formula (256*(32*board + port+1) + 23).

```
11.234.87.215 02/01 terminal02
```

The above example shows an entry for Port ID Via Telnet. telnetd will use this entlry to map the DTC's port and board numbers to the name being used for the connection on the HP-UX system.

Files

```
/etc/dpp
/etc/ocdebug
/etc/ocd
/etc/dpp_login.bin
/etc/utmp.dfa
/etc/newconfig/ddfa/pcf
/etc/newconfig/ddfa/dp
```

See Also

ddfa(7) dpp(1m) ocd(1m) ocdebug(1m) pcf(4)

dpp(1m)

Name

dpp - Dedicated Ports Parser, used by DDFA software

Synopsis

```
dpp <dp_file> [-l<log_file>] [-c] [-k] [-p<ocd_program>]
```

Description

This file is part of the HP Datacommunications and Terminal Controller (DTC) Device File Access (DDFA) software. It parses the Dedicated Ports file (dp) and calls the Outbound Connection Daemon (ocd) to spawn a daemon for each valid entry in dp.

dpp can be run from the shell or it can be included in netlinkrc to automatically run the DDFA software each time the system is booted. dpp can only be run by root.

Refer to ddfa(7) for information on how to configure and install the DDFA software.

The following arguments can be appended to dpp:

`<dp_file>` — This is mandatory, and must be the first field. The dp file must be executable and meet the specifications given in dp(4). The dp file defines the link between the DTC port and the device file used by applications to access the port. If the dp file is modified, dpp must be re-run on it to take account of the changes.

`-l<log_file>` — If the log file isn't specified, all error messages will be logged to the screen. If it is specified, messages will be logged to the file.

The file name must be specified, if it doesn't exist it will be created. The file must be non-executable and readable by dpp.

`-p<ocd_program>` The default path for ocd is /etc/ocd: if the path is different, it must be specified using the -p switch. The ocd must be executable with access rights for dpp.

dpp starts a process for each valid entry in the dp file and starts execution of ocd.

`-k` Removes the device file corresponding to each valid entry in the dp file, and then launches ocd for each valid entry.

The ocd program normally creates and removes devices files. However, if the process is killed badly, for example with kill -9, the device file may remain. If the system is rebooted, the -k option may be specified to restart all the dp file entries correctly. Deleting the device file will eventually cause the ocd process (if any) to exit.

If a corresponding ocd no longer exists, the device file will be removed by any following invocation of an ocd which requires the same device file.

`-c` This option parses the dp file and logs all bad entries. It is useful for debugging the dp file before running it properly. The -p<ocd_program> entry is ignored if -c is used.

The netlinkrc script can be modified to include dpp, for example:

```
/etc/dpp /etc/ddfa/dp -k
```

dpp runs, scans all the entries in the dp file, creates a process for each valid entry then exits. It is recommended that the -k option is used, and that a log file is specified.

If the dp file is modified, dpp must be re-run to take account of the changes.

Killing Daemons

Outbound Connection Daemons (ocds) should be killed using kill -15. kill -9 is not recommended as the device file remains. ocd will verify the validity of an existing pseudonym before trying to use it. dpp and ocd use data stored in the file /etc/utmp.dfa to verify whether a process still owns a pseudonym before taking it over. If ocd finds an unowned pseudonym, it will use it.

Errors

Error messages are logged for bad arguments, bad file entries and ocd process creation errors. By default, they are logged to stdout.

If the -l<error_log> option is used they are appended to the log file The error messages are generally self-explanatory, but a brief explanation of some has been included in the following list.

`error 0: <dp_file> is mandatory.`

`error 1: dp file must be the first argumnet.`

`error 2: Cannot read dp file.`

The dp file either doesn't exist or can't be accessed with current access right.

`error 3: No log file defined (-l option).`

`error 4: Cannot create log file (-l option).`

The log file cannot be created either because of an invalid path or because of insufficient access rights.

`error 5: Cannot access log file (-l option).`

The log file cannot be accessed, either because of an invlaid path or because of insufficient access rights. The log file must be readable by anyone.

`error 6: No ocd file defined in program option.`

`error 7: Cannot execute ocd program (-p option).`

The ocd program specified in the -p option either doesn't exist or is not an executable file with current access rights.

`error 8: Cannot purge device file (/dev/xxx).`

The -k option has been specified, the device file exists, but it cannot be purged because of insufficient access rights.

`error 9: Cannot execute default program (/etc/ocd).`

The default ocd cannot be executed, either because of insufficient access rights or because it has not been correctly installed.

`error 10: Entry ignored (Bad IP address).`

The dp file entry specified does not have a valid IP address.

`error 11: Entry ignored (no board/port info).`

`error 12: Entry ignored (Bad port number).`

The port specified is either not a decimal value or a string composed of x or X characters.

`error 13: Entry ignored (Bad board number).`

The board specified is either not a decimal value or is not a string composed of x or X characters.

`error 14: No more processes available on system.`

The ocd program specified cannot be started because there are no processes available on the system.

`error 15: Entry ignored (no device name).`

`error 16: Entry ignored (Bad device name).`

The device file specified cannot be created either because of an invalid path or insufficient access rights.

`error 17: Entry ignored (Bad config_name).`

The specified config file cannot be read either because of an invalid path or insufficient access rights.

Files

```
/etc/dpp
/etc/ocdebug
/etc/ocd
/etc/dpp_login.bin
/etc/utmp.dfa
/etc/newconfig/ddfa/pcf
/etc/newconfig/ddfa/dp
```

See Also

ddfa(7) dp(4) ocd(1m) ocdebug(1m) pcf(4)

pcf(4)

Name

pcf - Port Configuration File, used by DDFA software

Description

This file is used by the HP Datacommunications and Terminal Controller (DTC) Device File Access (DDFA) software to configure individual DTC ports. pcf is the generic name of the template file, in practice it is renamed for each port which needs different configuration values, and the values are altered appropriately for the device attached to the port. The pcf is referenced by an entry in the Dedicated Ports file (dp). The Dedicated Port Parser (dpp) parses the dp file and calls the Outbound Connection Daemon (ocd) program to spawn a daemon for each valid line in the dp file: a valid line is one in which the fourth field is the name of a pcf.

The master pcf is /etc/newconfig/ddfa/pcf, and should only be referenced in the dp file if the default values it contains are correct for the ports. If different values are needed, /etc/newconfig/ddfa/pcf should be copied to another directory and the copy should be modified and referenced in dp. We recommend that you create the directory /etc/ddfa to hold the pcfs and the modified dp file.

Refer to ddfa(7) for information on how to configure and install the DDFA software.

The file consists of the names of variables and their values. The variables are shown terminated by a colon (:), but this is not mandatory. A variable and its value can be separated by spaces or tabs. Only one variable-value pair is allowed per line. Only the value should be altered, the variable name should not be changed.

The file contains the following information:

`telnet_mode:` This can have the value disable or enable. When it is enabled, data transfer over the network will use the Telnet protocol. This option must be enabled for the DTC.

`timing_mark:` This can have the value enable or disable. When it is enabled, a telnet timing-mark negotation will be sent to the DTC after all user data has been transferred. ocd will wait for a reply to the timing mark negotiation before closing the connection. This ensures that all data has been output from

the DTC buffers to the device before the buffers are flushed. It should therefore be enabled for the DTC.

`telnet_timer:` This defines the time, in seconds, during which the software will wait for a response to the telnet timing mark and binary negotiation.

If the timer pops, an error message will be will be logged to /usr/adm/syslog and the error will be transmitted to the user application.

`binary_mode:` This can have the value disable or enable. When it is enabled, data transfer over the network will be in binary mode, and treatment of special characters (for example XON/XOFF) will be disabled.

Due to the absence of flow control, data integrity cannot be guaranteed when binary_mode is enabled.

Note that even if binary_mode is disabled, it may be negotiated at any time by the application setting IXON to 0 in the termio data structure.

`open_tries:` This defines the number of times the software will try to open a connection before giving up. If the value is 0 the software will try "forever" (approximately 68 years). If the retry process fails, an error message is logged to /usr/adm/syslog. The error message is also transmitted to the user application.

The retry process can be interrupted by sending the SIGUSR2 signal to the ocd process using kill -17 .

Note that if the application exits after asking ocd to open the connection to the DTC, ocd will continue trying to open until open_tries and/or open_timer are exceeded.

`open_timer:` This defines the time in seconds between tries. If the value is 0 ocd uses an exponential retry period algorithm up to 32 seconds, that is, 1 2 4 8 16 32 32 32 ...

`close_timer:` This defines the time in seconds between the close call made by the application on the pty slave and the moment when the connection is actually closed. Setting this value to, for example, 5 seconds avoids the overhead of opening and closing the connection when a spooler spools several files at a time. Setting a sufficiently high value effectively leaves the connection permanently open.

`status_request:` This may have the value disable or enable. When it is enabled, the software sends a status request to the device attached to the server and processes the reply as follows:

`LP_OK (0x30)`
cd continues processing.

`LP_NO_PAPER (0x31)`
cd retries within the limits of the status timer.

`LP_BUSY (0x32)`
cd retries within the limits of the status timer.

`LP_OFF_LINE (0x23)`
cd retries within the limits of the status timer.

`LP_DATA_ERROR (0x38)`
ocd retries within the limits of the status timer.

`status_timer:` This defines the time, in seconds, after which the software no longer waits for the reply to the status request. If the timer pops, an error message is logged to /usr/adm/syslog. The error condition is also transmitted to the user application.

`eightbit:` This can have the value enable or disable. Normally, data bytes processed by the pty have bit 7 stripped. If eightbit is enabled, the stripping is disabled. If eightbit is disabled, stripping is enabled, and bit 7 is stripped. This can also be achieved by changing the pseudonym's termio structure using the ioctl commands.

`tcp_nodelay:` This can have the value enable or disable If it is enabled, data is sent to the LAN as it is received. It can be disabled if the software is sending packets faster than the server can accept.

The default values are:

```
telnet_mode: enable
timing_mark: enable
telnet_timer: 120
binary_mode: disable
open_tries: 1500
open_timer: 30
close_timer: 0
status_request: disable
status_timer: 30
eightbit: disable
tcp_nodelay: enable
```

Files

```
/etc/dpp
/etc/ocdebug
/etc/ocd
/etc/dpp_login.bin
/etc/utmp.dfa
/etc/newconfig/ddfa/pcf
/etc/newconfig/ddfa/dp
```

See Also

ddfa(7) dp(4) dpp(1m) ocd(1m) ocdebug(1m)

ocd(1m)

Name

ocd - Outbound Connection Daemon, used by DDFA software

Synopsis

```
ocd -n<node_name> -f<pseudonym> [-b<board_no>] [-p<port_no>]
[-c<config_file>]
```

Description

The Outbound Connection Daemon (ocd) is part of the HP Datacommunications and Terminal Controller (DTC) Device File Access (DDFA) software. It manages the connection and data transfer to the remote DTC port. It may be spawned from the Dedicated Port Parser (dpp) or run directly from the shell.

For performance reasons, ocd does not have a debug mode: however, a version called ocdebug with debug facilities is available. See ocdebug(1m) for more information.

See ddfa(7) for more information on how to configure and install the DDFA software, and for an explanation of how it works.

The following arguments are allowed:

`-n<node_name>` Mandatory. This is the IP address of the terminal server or the port.

`-f<pseudonym>` Mandatory. This is the absolute or relative path to the device file which is linked by the software to the reserved pty. Applications use the pseudonym not the dynamically allocated pty slave.

`-b<board_no>` Optional. This refers to the board number of the DTC. If it is omitted, the port number option must contain the full TCP service port address. -b and -p must not be used if the IP address given in -n is the IP address of a port.

If the -n option explicitly names a DTC port, the -b option is not needed.

`-p<port_no>` Optional. This is the DTC port number. If the -b option is omitted, the port number must be the TCP service port address which will be used by the software to access the port. If the value is omitted, the value 23 (Telnet) will be used by default.

`-c<config_file>` Optional. This is the name (including the absolute path) of the configuration file used to profile the DTC port. If this value is omitted, the default values specified in the default pcf file (/etc/newconfig/ddfa/pcf) will be used.

If the pcf doesn't exist, an error message will be logged, and the following values will be used (note that the values for open_tries and open_timer are different from the default values):

```
telnet_mode: enable
timing_mark: enable
telnet_timer: 120
binary_mode: disable
open_tries: 0
open_timer: 0
close_timer: 0
status_request: disable
status_timer: 30
eightbit: disable
tcp_nodelay: enable
```

ocd logs important messages and error conditions to /usr/adm/syslog.

Files

```
/etc/dpp
/etc/ocdebug
/etc/ocd
/etc/dpp_login.bin
/etc/utmp.dfa
```

```
/etc/newconfig/ddfa/pcf
/etc/newconfig/ddfa/dp
```

See Also

ddfa(7) dp(4) dpp(1m) ocdebug(1m) pcf(4)

ocdebug(1m)

Name

ocdebug - Outbound Connection Daemon debug utility, used by DDFA software

Synopsis

```
ocdebug -n<node_name> -f<pseudonym> [-b<board_no>] [-p<port_no>]
[-c<config_file>] [-d<level>]
```

Description

The Outbound Connection Daemon (ocd) is part of the HP Datacommunication and Terminal Controller (DTC) Device File Access (DDFA) software. It manages the connection and data transfer to the remote DTC port. For performance reasons, it does not have a debug mode: ocdebug is a version of ocd with debug facilities. See ocd(1m) for more information on ocd.

See ddfa(7) for more information on how to configure and install the DDFA software, and for an explanation of how it works.

Debugging may be toggled interactively by sending the SIGUSR1 signal to the process by typing kill -16 <pid>.

The following arguments are allowed (note that apart from -d they are the same as the ocd arguments):

-n<node_name>	Mandatory. This is the IP address of the server or the port.
-f<pseudonym>	Mandatory. This is the absolute or relative path to the device file which is linked by the software to the reserved pty. Applications use the pseudonym not the dynamically allocated pty slave.
-b<board_no>	Optional. This refers to the board number of the DTC. If it is omitted, the port number option must contain the full TCP service port address.

If the -n option explicitly names a DTC port, the -b option is not needed.

`-p<port_no>` Optional. This is the DTC port number. If the -b option is omitted, the port number must be the TCP service port address which will be used by server to access the port. If the value is omitted, the value 23 (Telnet) will be used by default.

`-c<config_file>` Optional. This is the path to the Port Configuration File (pcf) used for profiling the DTC port. If this value is omitted, the default values specified in the default pcf file (/etc/newconfig/ddfa/pcf) will be used.

If the pcf doesn't exist, an error message will be logged, and the following values will be used (note that the values for open_tries and open_timer are different from the default values):

```
telnet_mode: enable
timing_mark: enable
telnet_timer: 120
binary_mode: disable
open_tries: 0
open_timer: 0
close_timer: 0
status_request: disable
status_timer: 30
eightbit: disable
tcp_nodelay: enable
```

`-d<level>` Optional. This indicates the level of debugging. The levels may be added together to accumulate dubugging functions. For example, -d7 will enable all levels, and -d3 will enable only the first two levels.

The levels are:

0 No debug messages.
1 Trace procedure entry/exit logged.

2 Additional tracking messages logged.
4 Data structures dumped.

The levels may be added together to accumulate dubugging functions.

Debug messages are logged to the file /usr/adm/ocd<pid>, and the file name is displayed at the start of the debugging.

Files

```
/etc/dpp
/etc/ocdebug
/etc/ocd
/etc/dpp_login.bin
/etc/utmp.dfa
/etc/newconfig/ddfa/pcf
/etc/newconfig/ddfa/dp
```

See Also

ddfa(7) dp(4) dpp(1m) ocd(1m) pcf(4)

Index

D

E

F

G

H

I

J

K

R

S

T

U

V

W

X

Edition 1, E0992
Printed in U.S.A. 09/92 English

Customer Order No.
B1014-90012

Manufacturing No.
B1014-90012
Mfg. number is for HP internal use only

B1014-90012