HEWLETT PACKARD

OSI Planning and Troubleshooting Guide

HEWLETT PACKARD

HP-UX 9000 Systems

OSI Planning and Troubleshooting Guide

Notice

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

HP-UX 9000 Systems

OSI Planning and Troubleshooting Guide

HEWLETT
PACKARD

Edition 3
E0392

Customer Order Number 32069-60001
Printed in U.S.A. 03/92

Using the HP OSI Documentation

Welcome to Hewlett-Packard's Open Systems Interconnection (OSI) networking. Use this manual to understand and plan your network layout, to define and create a network addressing scheme for configuring OSI products, and to verify interoperability with remote nodes. You should also use this manual to troubleshoot problems in the OSI stack with logging and tracing, and Hewlett-Packard's OSI diagnostic tool, *osidiag*.

Proceeding through the Documentation Set

As you implement your HP OSI products (for example, X.25/9000, OTS/9000, and FTAM/9000), you'll read chapters from both this manual and the appropriate product manuals. The product manuals contain information specific to those products. The *OSI Planning and Troubleshooting Guide* (the OSI manual) discusses topics related to the entire OSI stack.

For example, network addressing (discussed in this manual) pertains to links (LAN, X.25), transport and session (OTS), and services (X.400, X.500, FTAM, MMS). Remote interoperability procedures also deal with the entire OSI stack.

Proceed through the manuals in this order:

- Planning Your Network, the *OSI Planning and Troubleshooting Guide*. Read this chapter to prepare your network map, to determine what services and vendors you'll be networking, and to answer general questions about your network.
- Gathering configuration data, OSI product manuals. Read the appropriate product manuals to understand what information you need to collect from your local node and from remote nodes in order for you to configure your OSI products.
- Installing the OSI products, OSI product manuals. Read the appropriate product manuals for instructions on how to install the product software.
- Configuring and verifying the OSI products, OSI product manuals. Read the appropriate product manuals for step-by-step instructions and how to enter the configuration information you gathered into the system and how to verify that the information was entered correctly.

- Performing Remote Interoperability Procedures, the *OSI Planning and Troubleshooting Guide*. Read this chapter to verify connectivity between your local node and other nodes on the network, and to identify where in the OSI stack problems (if any) have occurred.
- Troubleshooting, the *OSI Planning and Troubleshooting Guide*. Read this chapter if you've run into an error while configuring and need general information on the troubleshooting process to isolate the problem to the failing product. Read the appropriate product manuals to troubleshoot the failing product.
- Logging and Tracing, the *OSI Planning and Troubleshooting Guide*. Read this chapter to prepare for troubleshooting by learning how to use the log and trace facility. You'll use the information in these files during the troubleshooting process.

Who Should Use This Manual

This manual is written for experienced HP-UX system and network administrators. Throughout this manual, "you" means the person or people responsible for the following:

- organizing the network within your domain
- issuing node names, addresses, and other parameters that are network unique
- installing the software on the local node
- maintaining the node based on configuration instructions

Depending upon the scope and complexity of your network, these responsibilities may be divided between "network administrators" and "system administrators." These responsibilities may also be performed by a single person. If your network environment includes separate system administrators and network administrators, review this material together to determine the division of tasks. For example, the system administrator might manage logging and tracing. The network administrator might want to customize specific products, like OTS/9000, to improve network performance.

In This Book

This manual, together with the product manuals for your HP OSI products, makes up the OSI documentation set. This manual provides the necessary information to understand network addressing, as well as how to test and troubleshoot network interoperability for the entire OSI stack. This manual is divided into nine chapters.

Chapter 1, "Hewlett-Packard's Open System Interconnection," introduces you to the OSI reference model, the various layers of OSI, and the relationships to ARPA and Internet products.

Chapter 2, "Planning Your Network," discusses creating a network map, determining what vendors and services will be on your network, and planning your addressing scheme; and explains general considerations for setting up your network.

Chapter 3, "Performing Remote Interoperability Procedures," contains step-by-step instructions for all the OSI products to test the communication connections between your local node and another node on the network.

Chapter 4, "Troubleshooting," describes general procedures for troubleshooting local configuration problems, as well as explanations of the tools you'll use to troubleshoot.

Chapter 5, "Using OSI Tools," describes the various tools used to configure, maintain, and troubleshoot HP's OSI products.

Chapter 6, "Logging and Tracing," discusses the uses of the logging and tracing facility, *nettl*, and shows examples of log and trace files.

Chapter 7, "OSIDIAG," is a detailed description of *osidiag*, the tool you use to diagnose and correct OSI problems.

Chapter 8, "Messages," contains error messages that may be encountered while using HP's OSI products.

Chapter 9, "Addressing," discusses general concepts about network addressing, how NSAPs are structured, and various addressing standards in use.

Contents

Chapter 6 Logging and Tracing

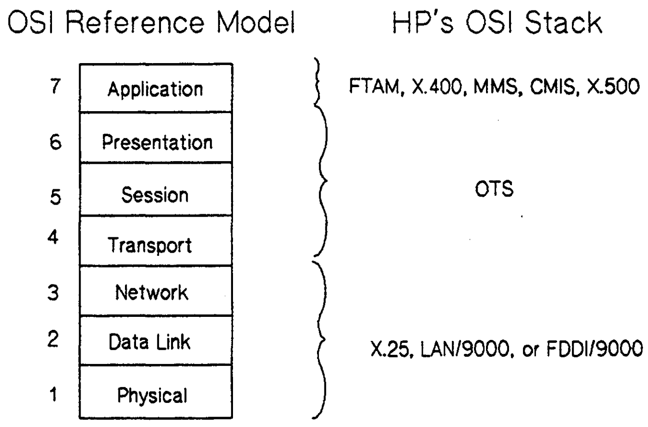
1

Hewlett-Packard's Open System Interconnection

An overview of Hewlett-Packard's OSI solution

Hewlett-Packard's Open System Interconnection

The HP-UX/OSI solution is a composite of OSI products that comply to the ISO model. The following diagram shows the HP products that provide the specific functions of each defined layer.



HP OSI Functions Defined by Layers

Hewlett-Packard's X.25 and LAN products perform the functions outlined by levels 1, 2, and 3 of the ISO model. X.25 supports wide area networks (WAN). LAN (802.3 or FDDI) support local area networks (LAN) functions.

OSI Transport Services (OTS) provides the functionality of levels 4, 5, and 6. These levels offer the necessary foundation to run the upper layer (7) applications and services such as HP FTAM, HP MMS, HP X.400, and HP X.500 Distributed Directory. OTS also provides application program interfaces (APIs) for the following levels:

- presentation layer - ACSE/Presentation and ROSE Interface (APRI)
- session layer - Session Interface
- transport layer - X/Open Transport Interface (XTI)

The File Transfer, Access, and Management (FTAM) application provides the capability to manipulate data files locally and at remote databases.

HP X.400 (1988 standard) is the interface for electronic messaging applications over a network. It allows you to exchange electronic mail with other users and perform message handling of other systems (i.e. SendMail to DEC All-in-1) through X.400 in a multivendor environment. HP X.400 also provides a high-level, easy-to-use application programmatic interface (API).

HP X.500 Distributed Directory (1988 standard) allows different vendors to store and gain access to a directory (a collection of information) on different systems.

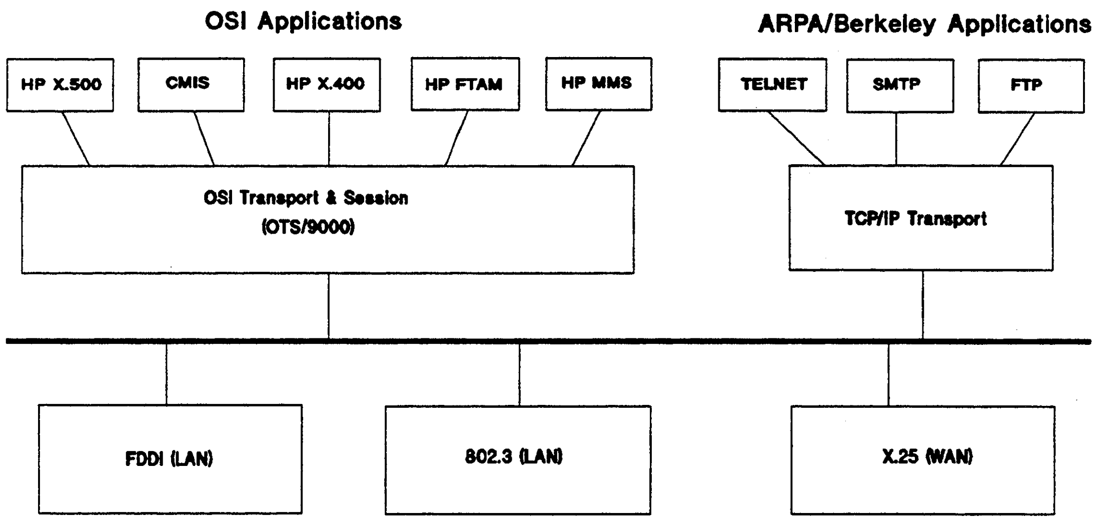
The Manufacturing Message Specification (MMS) application provides the user with the capability to control and coordinate programmable factory floor devices involved in manufacturing.

The Common Management Information Service (CMIS) is the interface for development of network management applications.

By supporting international standards specified by International Standards Organization (ISO) and Consultative Committee for International Telegraphy and Telephony (CCITT), HP's OSI products interoperate in a multivendor environment.

The HP OSI stack and services can run independently or share the HP-UX processor with ARPA services. The dual protocol host configuration allows OSI and ARPA services to coexist on existing networks. Local (802.3) and Wide Area (X.25) network link cards are shared by both protocol stacks.

The figure below shows the HP OSI software products in relation to the ARPA and Internet products.



2

Planning Your Network

The process you should use to successfully bring up a new OSI network or add to an existing OSI network

Planning Your Network

This chapter describes the process you should use to successfully bring up a new OSI network or add to an existing OSI network.

The following steps should be taken to get your OSI network up and running:

- determine the services required
- determine the vendors involved
- determine the network structure
- determine your addressing scheme
- gather and distribute configuration information
- install and configure systems
- start up systems (if needed)
- perform local verification and interoperability tests

The following sections describe each step in more detail and in some cases direct you to other chapters for further information.

Determine the Services Required

The first step in planning your network is to identify what communication problems you are trying to solve and what OSI service or services best address your needs. This section describes each service provided by Hewlett-Packard and when it may be appropriate for meeting your communication needs.

X.500

X.500 provides a directory service. The directory allows you to look up attributes associated with a symbolic name. For instance, you could look up the address of a remote node based on that node's name, or look up the mail location of a user.

The directory can save configuration time as well as provide greater data integrity by keeping information like that described above in a central location, as opposed to being redundantly stored on each node in the network.

Each node gaining access to the X.500 directory must be able to act as a Directory User Agent (DUA). In addition, at least one node on the network must act as the Directory Server Agent (DSA). The DSA stores the actual information, whereas the DUAs post queries for information. HP supports both facilities.

Use X.500 if a large amount of addressing information (either for network nodes or mail users) is required by the nodes on your network.

X.400

X.400 provides a message handling service, or more simply, electronic mail. It allows you to encapsulate electronic information and reliably deliver it to one or more destination users.

X.400 messages are transferred in a store and forward manner; hence X.400 is inappropriate for transferring data in real-time or other time critical applications.

Use X.400 if electronic messaging is required between users of the OSI network.

CMIS

CMIS allows management of "managed objects" over the OSI network. Managed objects are abstractions of actual resources located on nodes in your network.

Through CMIS you can enable and disable network resources, view and change configurable values and register for notification of exceptional events.

A system supporting CMIS may act only as a requestor of operations against a resource (an agent), or only as a responder to such requests (a manager), or both. It is important that you understand what resources you want managed and if this capability is available through a particular CMIS implementation.

HP provides a programmatic interface to CMIS, which allows you to write applications acting as both a requestor or a responder. However developers are responsible for making actual HP resources appear as CMIS objects through their applications.

Use CMIS if distributed management of nodes and other network resources is desired.

FTAM

FTAM allows you to transfer files across an OSI network. It also provides facilities to create, delete and modify files on a remote node.

HP's FTAM provides both a programmatic and interactive interface to FTAM. For routine file transfers, the interactive interface makes this task very easy. For more complex or automated file transfers, the programmatic interface is available.

Use FTAM if you wish to move or manage files across an OSI network.

MMS

MMS provides you with interprocess messaging capabilities, with specific messages geared towards controlling manufacturing devices (for example, Programmable Logic Controllers, Robots, and Numeric Controllers).

The operations include the ability to control programs remotely, send structured data via the Variable Management facility, and to transfer files.

HP provides access to MMS through the Manufacturing Automation Protocol 3.0 (MAP 3.0) API. This standard interface allows you to use 39 different MMS primitives as both initiator and responder.

Use MMS if you are communicating with factory floor devices which currently support MMS. You can also use MMS for general Interprocess Communication needs.

APRI

The ACSE/Presentation and ROSE Interface (APRI) provides you with the ability to implement your own OSI application layer protocols, or custom applications. The facilities provided by ACSE and Presentation are a general interprocess communication mechanism, with the addition of Abstract Syntax Notation One (ASN.1) context negotiation.

ROSE provides remote operation capabilities to invoke operations on the remote and receive status on completion. This service runs on top of the service provided by ACSE and Presentation.

Use APRI for porting layer 7 protocols to the HP platform, for developing specialized layer 7 applications, and for remote operation applications.

Session

This interface provides you with access to layer 5 of the OSI stack. The Session layer provides the ability to perform interprocess communication, as well as several functions to assist in the establishment of checkpoints and recovery in the data stream.

The Session layer is the lowest layer that provides graceful release of connections, that is, guaranteed arrival of all data before termination of the connection. Transport only has abrupt release.

Use Session for developing IPC applications which do not require the ASN.1 context management facilities of APRI, and cannot accept the overhead of MMS. Applications that do require graceful release or data stream checkpoint and recovery, should use Session rather than XTI.

XTI

XTI is an X/Open standard interface to the Transport layer. The Transport layer provides a very simple set of functions, for example, connect, send data (normal and expedited) and disconnect. These building blocks provide a basic and low overhead access to performing interprocess communication over an OSI network.

The use of a standard interface makes your application source code portable between platforms following the standard.

Use XTI for porting existing IPC applications that run over non-OSI networks and for achieving high performance as a result of low protocol overhead.

Determine the Vendors Involved

Another factor in determining the ultimate layout of your network is the equipment various vendors use to communicate.

Not all vendors provide each OSI service. Also, a given vendor may only provide some subset of functionality for a given layer. For example, HP does not currently provide the Virtual Terminal service. Another vendor may not provide X.500. Or another vendor may provide a Network Management Agent, but have no facilities to manage the resources local to that node.

When planning your network, you need to take such factors into consideration. Below is a list of items you should investigate to help ensure your success.

- Does the vendor provide this service?

  If you wish to communicate with a piece of equipment, it must support the service. Access to mid-stack layers (Session, Transport) cannot be assumed just because a vendor provides a seven layer stack. Verify that access is provided if you intend to use mid-stack access.

- Does this service coexist with other facilities I want to use on this node?

  Determine what effects installing this product has on the other operations of your system. Especially check on its coexistence with other non-OSI networking services you may be using.

- What version of this service is provided?

  Some standards are evolving. For example, MMS products may support the Draft International Standard (DIS) or the International Standard (IS) or both. Session provides both Version 1 and Version 2. Verify that the version supported is acceptable to all nodes you will communicate with.

- What functional units of this service are provided?

  Many OSI implementations are a subset of the full service. You should understand what facilities you require from this service and ensure that each vendor supports those.

- What lower layer protocols does this product use? What versions are these? Can I use it over the link I want?

  In addition to verifying that the implementations of the service you are using are compatible, you should also verify that the underlying layers are compatible. The primary considerations are Session version and functional units, Transport Layer class, Network Layer protocol and link type. More detailed questions about the links used are contained below in "General LAN Questions" and "General X.25 Questions."

- What is the interface to this product? Is it standardized? Does it meet my needs?

  The interfaces to the OSI services can be broadly broken into two categories: interactive, where you type commands, or select choices from a menu; and programmatic, where you develop your own program (typically in C) using a set of OSI library commands. Some standard interfaces are emerging. MAP and X/Open have defined interfaces (both programmatic and interactive) to various services. Support of a standard interface will increase the portability of your applications and make the transition from one platform to another simpler.

- What level of conformance or interoperability testing has been achieved by this product?

  Conformance testing indicates that an implementation has successfully passed a suite of tests against a reference OSI implementation. Such testing improves the chances that the implementation is correct and that it will interoperate with other conformant systems. Interoperability testing is a direct test between two vendors' implementations. Having this testing done in advance of your installation can save you a lot of time. Ask your HP Support Representative for an up-to-date list of systems which interoperate with HP systems.

- Who will install, configure and test this product?

  If you will not be doing the entire installation yourself, you should coordinate with the other individuals involved. HP provides a remote system worksheet which you can use to gather information for these other systems.

General LAN Questions

- Is ES/IS (end system/intermediate system) protocol supported?

  Most systems today support the ES/IS protocol. This protocol allows systems to dynamically maintain information about the NSAP to MAC address mapping for systems on a LAN. By using this protocol, a great deal of configuration can be avoided.

- Can this system behave as an IS?

  An intermediate system is capable of forwarding traffic from one subnetwork to an IS on another subnetwork. This system should support both the ES/IS and IS/IS protocols. HP systems do not behave as an IS.

- Will this subnetwork use the NULL subset of CLNS?

  The NULL subset of CLNS is used for systems that are on an isolated LAN network. When NULL subnetting is used, no traffic can be forwarded via routers. The benefit of the NULL subnet is reduced protocol packet size (on the order of 40 bytes per packet). The drawback is the inability to communicate outside the subnetwork.

General X.25 Questions

- Is ISO 8878 behavior supported?

  ISO 8878 defines how X.25 is to be used to provide CONS. Among other features, it defines a mechanism for conveying NSAPs in 1984 and 1988 X.25 CALL packets. It also defines a "special" mechanism, called SNDCP, which simulates this behavior on 1980 X.25 networks. OTS supports full 8878 by default. However, many OSI implementations do not support the SNDCP behavior. The configurable parameter, "snet_allow_iso8878" in the file "ots_subnets" controls whether or not OTS will use the SNDCP when communicating over 1980 X.25. See the *Installing and Administering OSI Transport Services* manual for more details about this parameter.

- What versions of X.25 will be used?

  There are two versions of X.25 support: X.25 '80 and '84. Each is effectively a superset of the previous. Sending X.25 '84 traffic to an X.25 '80 system may result in connection refusal. It is important to understand what systems support which protocol. OTS by default sends X.25 '84 packets and receives both X.25 '80 and '84. See the *Installing and Administering OSI Transport Services* manual for a discussion of the configurable parameters "snet_allow_x25_1980" and "snet_allow_x25_1984" in the file "ots_subnets" for more information.

- What level of X.25 can my switch or PDN use?

  If your switch or Public Data Network only supports X.25 '80, then systems connected through it must use X.25 '80. Otherwise '80 and '84 are acceptable.

- Is subaddressing supported?

  HP recommends the use of subaddressing in order to share the same physical card with various services (see the discussion of subaddressing in chapter 9 of this manual). If subaddressing is not supported, then you will be restricted to configuring a single CONS and a single CLNS network through each card. See the *Installing and Administering OSI Transport Services* manual for a discussion of the parameter "snet_bind_by_pid" in the file "ots_subnets".

- Are Protocol ID values used?

  This question is important if subaddressing is not allowed by the network. If the protocol ID field (sometimes referred to as "Call Used Data") is carried on each CALL packet sent by the remote, then you can safely set the "tpcons_null_pid" parameter. OTS by default sends the PID. The parameter "tpcons_null_pid" in the file "ots_parms" can override this behavior. By default, OTS will also accept call packets with or without a PID value. Setting the "snet_bind_by_pid" parameter in the "ots_subnets" file will prevent OTS from accepting CALL requests without a PID. See the *Installing and Administering OSI Transport Services* manual for a discussion of this parameter.

- What Network layer protocols will be used (CONS, CLNS)?

  The Connectionless Network Service (CLNS) has the advantage in that it can be used with routers and does not require the gateway facility provided by MSDSG. However, CLNS does present more overhead because each packet sent contains addressing information, and Transport Class 4 must be used. The Connection-Oriented Network Service (CONS) is currently more widely used than CLNS/X.25.

General Network Questions

- Will I be stringing new cable for OSI or using existing infrastructure?

  Your OSI network should be capable of coexisting with other network traffic over existing LAN or X.25 cable plants. (See the question, "Does this service coexist with other facilities I want to use on this node?" in the section "Determine the Vendors Involved.") If you do not have an existing cable plant, you will need to plan your cabling.

- Will existing repeaters, bridges, routers support OSI traffic?

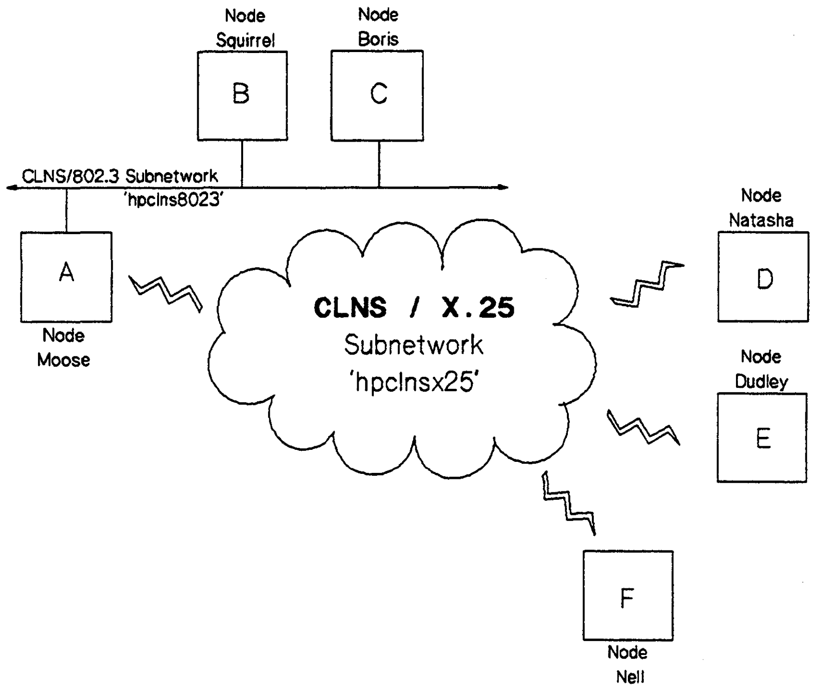
  If you intend to use existing network equipment from a non-OSI network, you should ensure that it will behave as expected. LAN repeaters and bridges should forward your traffic, assuming they support IEEE 802.2 (which they should). Routers which understand only IP (MIL-STD-1777) and not OSI CLNP, ES/IS and IS/IS will not forward your traffic.

Determine the Network Structure

At this point, you have identified the nodes and the services you will be using for your network. The next step is to draw a network map to visualize the structure of your network.

If this is an addition to an existing network, you should update the existing network map (if one exists). Otherwise, you should follow these steps to draw your map. A simple example of a network map is shown below.

- Identify the subnetworks and nodes in your network.
- Assign unique names to each node and subnetwork and other pieces of network equipment.
- Draw the subnetworks, using lines for LANs and clouds for X.25.
- Place each node on the subnetworks you've drawn.
- Place any other pieces of network equipment on the diagram (for example, routers, repeaters, X.25 switches, and analyzers).
- Identify any non-OSI nodes or network equipment which may be sharing the network.



Note Information beyond the node name, such as X.25 subaddresses, network addresses, and applications, can be kept separately by making copies of the "Remote System Worksheet" contained in the *Installing and Administering OSI Transport Services* manual. This is described in the subsection "Gathering Configuration Information."

Determine the Addressing Scheme

Network Address

At this point, you have determined the layout of your network. You should now develop an addressing scheme that facilitates routing in your existing network, as well as future expansion.

Note Chapter 9 in this manual provides a detailed discussion of network address formats. Unless you are thoroughly familiar with network addressing, read chapter 9 before proceeding.

Most network address formats allow you to segment your network into routing domains and areas. Examine your network map. Areas should be defined as groups of subnets that are connected and have common bandwidth capabilities. For example, if you have two LAN subnetworks connected by an Intermediate System, you may consider them to be in the same area. You should not assign the same area ID to subnets that are connected to one another, but differ in bandwidth (for example, an X.25 subnetwork and a LAN subnetwork, or two LANs connected via a modem link).

One other consideration with areas is that they should not contain more than 10,000 nodes. This is dictated by the IS to IS protocol.

The assignment of domain identifiers is largely a policy decision. Any number of areas may be assigned to a single domain. However, a limitation of 10,000 areas per domain is a good rule. You may assign a single domain ID to your entire network unless the following criteria are met:

1. Any area that is isolated from another area should have a different Domain ID.

2. If, for security reasons, you wish to distinguish between traffic in different areas, you may want to assign different domain IDs to those areas. For example, you might assign "Finance" and "Research" different domains.

After you have done the partition into domains and areas, you can then assign an appropriate network address prefix for each node, and the full address can be completed by using each node's End System address.

Application Addresses

You may also want to dictate the selector values to be used for the upper layer addresses. (Note that HP OSI services use default addresses. For instance, FTAM uses 0x0001 (hex) for P-, S-, and T- selectors. X.400 uses blank (null) P- and S- selectors and 4D4853 (hex) as the T-selector. Refer to the appropriate product manuals to see what, if any, default addresses are used.)

If you wish to change the default addresses, you might assign all FTAM responders to have the P-, S- and T-selectors of 0x0010. For an XTI application you are developing, you may want all initiators to use T-selectors 0x0020 and all responders to use T-selectors of 0x0021.

You have a wide array of addresses to choose from. The selectors may fall in the following range:

P-selector: 0 to 16 bytes

S-selector: 0 to 16 bytes

T-selector: 0 to 32 bytes

Remember that the longer or more complicated you make your selector values, the more room you leave for error during configuration.

Note Some government profiles may dictate the selector values for certain services (for example, U.S. GOSIP for FTAM and X.400).

Naming Hierarchies

If you are using FTAM, MMS, or X.500, you will use Directory Distinguished Names (DDNs) in some capacity. These are hierarchical names, for example, "/C=us/O=hp/OU=hpnode1/AP=mms/AE=demo_prog".

At this time, you should define what components of the naming hierarchy you wish to use and define conventions for assigning values to each component.

As an example, you might decide that the Country attribute is not to be used, the Organization attribute will always be your company's name in all lower case, and the Organization Unit will correspond to the node name given on the network map in all lower case.

See the discussion of naming in the configuration manuals for the respective services for more information about what attribute classes are available and which ones are required.

Gather and Distribute Configuration Information

At this point, you have determined your network topology and addressing scheme for your network. You are now ready to compile the configuration information required for each system so that they may communicate with one another.

HP has provided a worksheet for gathering information about each node on the network. It is contained in the *Installing and Administering OSI Transport Services* manual in chapter 1. (You may also need additional configuration information for other HP OSI products, for example, FTAM and MMS. See the appropriate product manuals for more information.)

After performing the data gathering steps described in chapter 1 of the *Installing and Administering OSI Transport Services* manual, you will be able to configure your HP system to communicate with each system on your network. The worksheet will also be useful for other vendors' network administrators.

Install and Configure Systems

The installation and configuration process for HP systems is described in the *Installing and Administering OTS* manual, as well as the manuals for the specific services (for example, FTAM and X.400).

Start Up Systems

The start up process for HP systems is described in chapter 5, "Using OSI Tools," as well as the manuals for specific services (for example, OTS, FTAM, and X.400).

Perform Local Verification and Interoperability Tests

The local verification steps for HP systems are described in the installation manuals. Interoperability testing procedures are described in the next chapter of this manual.

3

Performing Remote Interoperability Procedures

The process for verifying and troubleshooting communication between your local HP node and another node on the network

Performing Remote Interoperability Procedures

This chapter describes the process for verifying and troubleshooting communication between your local HP node and another node on the network. This chapter assumes the OSI products are installed, configured, and verified using local loopback tests. (Refer to the OSI product manuals for details.) The verification and troubleshooting procedure is broken into eight components:

- **X.400** - This component verifies the ability to transfer X.400 mail messages between systems. If these tests pass, there is no need to verify the lower layers. If problems are encountered, you may need to perform lower layer verification and troubleshooting.
- **MMS** - This component verifies the ability to establish an MMS connection with a remote MMS application. If these tests pass, there is no need to verify the lower layers. If problems are encountered, you may need to perform lower layer verification and troubleshooting.
- **FTAM** - This component verifies the ability to connect with and transfer files to the remote FTAM Responder. If these tests pass, there is no need to verify the lower layers. If problems are encountered, you may need to perform lower layer verification and troubleshooting.
- **ACSE/Presentation and ROSE** - This component verifies the ability to connect through the APRI Services. If these tests pass, there is no need to verify the lower layers. If problems are encountered, you may need to perform lower layer verification and troubleshooting.
- **Session** - This component verifies that the ability to establish a Session connection with a remote Session application. If these tests pass, there is no need to verify the lower layers. If problems are encountered, you may need to perform lower layer verification and troubleshooting.
- **OTS** - This component verifies that the local and remote stacks can communicate at the Transport layer. If these tests pass, then the link products will have been verified.
- **LAN** - This component (802.3 or FDDI) verifies that the local node can pass data frames to the remote across the LAN. Problems at this layer typically involve hardware or cabling problems.
- **X.25** - This component verifies that the local node can establish X.25 connections with the remote node. Problems at this layer typically involve hardware or configuration problems.

Testing X.400 Interoperability

The steps below describe how to test X.400 connectivity between the local and remote node. See the "osidiag" section in chapter 5 this chapter for an overview, or chapter 7 for a detailed discussion about *osidiag*.

Pre-Test Checklist

1. Ensure that X.400 and OTS are running on the local system.
2. If using X.400 with HP Desk, ensure that HP Desk and HP X.400/HP Desk software are up and running on the HP 3000.
3. If using X.400 with mailx or ELM, ensure that Sendmail is up and running on the local system.
4. Ensure that X.400 and the OSI stack are up and running on the remote system.
5. Do "X.25 Interoperability Testing" (if applicable). Refer to "X.25 IOP Testing" later in this chapter for more information.
6. Do "LAN Interoperability Testing" (if applicable). Refer to "LAN IOP Testing" later in this chapter for more information.
7. Do "OTS Interoperability Testing." Refer to "OTS IOP Testing" later in this chapter for more information.

User Agent and Message Transfer Agent (MTA) Tests

1. Read the various sections of this manual on X.400 Tests and the method to interpret the results of the test.
2. Start *osiadmin* and select X.400.
3. Start X.400.
4. Select "Status" from the X.400 screen in *osiadmin*.
   - Check if event logging is enabled for RTS, MTA. If using X.400 with HP Desk, check if event logging is enabled for encoder, decoder, and *x4xfer*. If using X.400 with mailx or ELM, check if recent logging is enabled for *x4mailer*. If any of the eventlogs are disabled, press (f3) and use "x4eventlog -e" to enable X.400 eventlogging.
   - Check if all X.400 components and OTS are running.
5. Select "Test Connectivity" from the X.400 menu in *osiadmin*.
6. Select "Utilities."
   - Select "Wait Time." This is the approximate time you expect to wait before a delivery report is received from the remote node. This is set to a default of 30 seconds. If you want the test to wait longer before checking if a delivery report has arrived, you should enter the correct time and press done.
   - The MTA test traces the message through the MTA, and RTS event logs or the encoder, decoder, *x4xfer* event logs if using HP Desk or *x4mailer* event logs,if using X.400 with mailx or ELM. If you are not familiar with HP's X.400 product, you could save the results from this test in a log file to study the output later. To save results, select "OPEN Result File" and enter the file name where the result must be saved.
   - Exit the Utilities section.
7. Select "User Agent Mail Test." This test sends a mail message to the remote User Agent, traces the message through the various log files and tracks the incoming delivery report corresponding to the message just sent. The various sections of this test are briefly mentioned below:
   - Prepare and send the message from the encoder if the HPDesk gateway option is selected, or mailx, if the mailx option is selected.
   - Track the outgoing Message through MTA and RTS.
   - Track the incoming Delivery Report. See chapter 7 for more information.

8. Enter the subject of the message to be sent.

9. Enter any ASCII filename you would like to send as the content of the message.

10. If you are using X.400 with HP Desk, enter "H" for "Product to Test," so that the test message will be sent from the encoder with *x4buildiff.* If you are using X.400 with mailx or ELM, enter "M" for "Product to Test," so that the message will be sent from mailx.

11. Press "Done."

12. If you entered "H" for "Product to Test," enter the Originator's O/R address and press "Done."

13. Enter the Recipient's O/R address for the message to be sent and press "Done" to start the test. The test echoes the information that was entered, such as the subject, originator, and recipient names.

    The test then displays "TRACKING OUTGOING MESSAGE." The output shows the progress of the message through the various log files. Each event logged has a eventlog number associated with it. To get more information on the significance of each of these messages, use the *x4solve* troubleshooting tool. Refer to the *Installing and Configuring HP X.400 Administrator's Guide* for more information on *x4solve*.

14. The important events to observe in this phase of the test are:

    - "Messages not found in RTS or MTA eventlog files." Refer to the "Common X.400 Errors" section for more information on these messages.
    - Non-delivery reports being generated by the MTA. This is indicated by the keyword "NON_DELIVERED" and a reason is printed in the line below. To get more information on the reason for non-delivery refer to the *Installing and Configuring HP X.400 Administrator's Guide* for MTA eventlog messages. Use *x4solve* to get more information on eventlog messages.
    - If the MTA successfully transferred the message to the RTS, but the RTS was unable to transfer the message to the adjacent MTA, this test has failed and you need to perform the RTS IOP tests. Once those tests have passed, rerun this test.
    - For any other error message generated in this stage, refer to the "Common X.400 Errors" section for more information.

15. The last phase displays "TRACKING INCOMING DELIVERY REPORT."

- If a delivery report is received, then the test passed. Proceed to step 16.
- If no delivery report arrived, check the "Common X.400 Errors" section of this chapter for action to take in this case.
- If a Non delivery Report arrived, then the test failed. Check the Reason for failure reported in the Non-delivery Report. Proceed to the "Common X.400 Errors" section later in this chapter for actions to take in this case.

16. If you are using X.400 with HP Desk, check the HP Desk node to see if the delivery report arrived in the originator's "In Tray." If you are using X.400 with mailx or ELM, check the mailbox of user "root" to see if the delivery report has arrived.

    If you are using X.400 with mailx or ELM, once you have received the delivery report from the remote node, your IOP testing is complete. If you are using X.400 with HP Desk, perform the next two tests from the HP X.400/Desk Node.

HP Desk to Remote Node Connection with HP as Initiator

1. Log on to the HP3000 as MGR.HPOFFICE.
2. Run *x400bit* to send a message to the remote system. See the *HP X.400/HP Desk Node Administrator's Guide* for more information on *x400bit*.
3. Run *x400util* to look at the HP X.400/HP Desk log and check if the message got out of the HP X.400/HP Desk. See the *HP X.400/HP Desk Node Administrator's Guide* for more information on *x400util*.
4. If the message is not successfully transferred to the X.400 Server, determine the problem by using the messages displayed by X400UTIL.
5. Check if the message was successfully transferred to the remote node. If it was successfully transferred, go on to test "HP Desk to Remote Mode with HP as Recipient."
6. If the message did not arrive at the remote node
   - Check the filename to which the message was copied when sent to the X.400 Server. This is in the log messages displayed by *x400util*.
   - Run *x4msgtrack* on the X.400 server with this filename.
   - Check the event displayed. For some of the error messages displayed, more information is available in the "Common X.400 Errors" section. More information on eventlogs is available in the *Installing and Configuring HP X.400 Administrator's Guide*. Use *x4solve* to get more information on eventlog messages.
   - If the RTS could not transfer the message to the adjacent MTA, this test has failed and you need to perform the RTS IOP tests. Once those tests have passed, rerun this test.

HP Desk to Remote Node Connection with HP as Recipient

1. Check if X.400 is running.
2. Send a message from the remote node to local HP Desk user.
3. If the message reached the local user, IOP testing is complete. If not, proceed further.
4. Run `/usr/lib/x400/x4logcat rts` to view the RTS event logs. Check if the incoming event was logged in the following file:

   `/usr/spool/x400/log/r<adj. MTA number>. evnt`

   If no event is logged in the incoming event file and event logging is enabled for RTS, then this test has failed and you need to perform the RTS IOP tests. Once those tests have passed, rerun this test.
5. Run `/usr/lib/x400/x4msgtrack` with the INCOMING file name to track the incoming message. Observe the results, and continue troubleshooting based on the events displayed.
6. If the message left the X.400 server successfully, run *x400util* on the HP 3000 side.
7. Observe the results and check if the message reached the X.400 system. If the message never reached the HP 3000 side, it must be a NS problem. Refer to the *HP X.400/HP Desk Node Administrator's Guide* for more information on the problem.

Common X.400 Errors

1. Some other invocation of *x4uatest* is running.

   Action : Remove all files from the /usr/spool/x400/tmp directory. Try again later.
2. When using X.400 with Sendmail: Failed to get File in *x4mailer* log.

   Action : Find outgoing message.

   - Make sure *x4mailer* is running (select STATUS on the X.400 test menu).
   - Make sure x4mailer event logging is on (select STATUS on the X.400 Test Menu).

- Make sure Sendmail is running (select STATUS on the X.400 Test Menu). Check the log file `/usr/spool/mqueue/syslog` for Sendmail log messages. Make sure that Sendmail configuration has been modified to work with X.400. See *Installing and Configuring X.400* for more information.

3. When using X.400 with HP Desk: Failed to find outgoing message file in encoder log.

   Action:

   - Make sure the encoder is running (select STATUS on the X.400 Test Menu).
   - Make sure the encoder event logging is turned on (select STATUS on the X.400 Test Menu).

4. Message not found in MTA eventlog.

   Action:

   - Make sure the MTA is running (select STATUS from the X.400 Test Menu).
   - Make sure the MTA event logging is enabled (select STATUS from the X.400 Test Menu).

5. Message not found in RTS eventlog.

   Action:

   - Make sure RTS is running (select STATUS from the X.400 Test Menu).
   - Make sure RTS eventlogging is enabled (select STATUS from the X.400 Test Menu).
   - Check MTA eventlog to see if a non-delivery report was sent. If a non-delivery report was sent, check the MTA eventlog for reasons why a non-delivery report was generated. The most common reasons are either no route configured for this address, or the wrong address was specified by the user.
   - Read the RTS eventlog corresponding to the adjacent MTA. This is the file `/usr/spool/x400 r<adjacent MTA num>.evnt`. The adjacent MTA number is logged by the MTA when it relays a message to the RTS. Check the time stamp on the last event to see if it was later than the time the message was sent. If it was later than the message sent time, proceed to the next item.
   - Check the last few events logged in the RTS eventlog file mentioned in item 3 above. Use *x4solve* to get more information on the eventlog messages.

6. The delivery report did not reach LOCAL MTA (was not in MTA log file).

   Action:

   - Make sure RTS and MTA are running. If you are using X.400 with HP Desk, make sure encoder, decoder and *x4xfer* are running. If you are using X.400 with Sendmail, Mailx, or ELM, make sure *X4mailer* is running (select STATUS from the X.400 Test Menu).
   - Make sure RTS and MTA event logging are on (select STATUS from the X.400 Test Menu).
   - Read the RTS eventlog corresponding to the adjacent MTA. This is the file `/usr/spool/x400/r<adjacent MTA num>.evnt`. Check the time stamp on the last event to see if it was later than the time the message was sent. If so, go to the next item below; otherwise, some problems with OTS or with connecting to the remote RTS are indicated. Use RTS and OTS Interoperability tests in *osiadmin* to troubleshoot further.
   - The following is an example logged event.

```
*******************************************************
06/08-16:58:13 INCOMING TRANSFER STARTED.  (X4EVENT 3523)
          Incoming MTA Name:  HOLLY831
          Incoming Password:  secret
06/08-16:58:14 INCOMING TRANSFER OF 434 BYTES SUCCESSFULLY COMPLETED.
         /usr/spool/x400/iq/M0608165814a.000  (X4EVENT 3524)
*******************************************************
```

     The important thing to observe is the Incoming File name (`/usr/spool/x400/iq/M0608165814a.000`). If these events are observed, proceed to the item below that begins, "Observe the output...", or else refer to the *Installing and Administering HP X.400 Administrator's Guide* for actions to take for the last few events logged.

   - Run *x4msgtrack* with the Incoming Filename `/usr/lib/x400/x4msgtrack -f` ***`filename`***. Check the MTA eventlog in the output to see if the delivery report arrived. This may happen if the "Wait Time" specified was too little. The delivery report is logged as a "DR-MPDU" and the time stamp on the logged message must be later than the time the message was sent. If the delivery report arrived, repeat UA Tests with a longer wait time. If the delivery report did not arrive, proceed further.
   - Observe the output and see if any error messages are logged. Check `/usr/spool/x400/iq` to see if the message is stuck there.

7. When using X.400 with Sendmail, mailx, ELM, the delivery report did not reach *x4mailer*.

   Action:

   - Make sure x4mailer and the MTA are running (select "STATUS" from X.400 test menu).
   - Check if *x4mailer* and MTA eventlogging are enabled (select STATUS from X.400 Test menu).
   - Check which output queue the MTA routed the message to (in the MTA eventlog). It should be oq253. If it isn't, either the originator address in the message was wrong or the route to the Sendmail connection is not configured correctly.

8. The delivery report did not reach the decoder.

   Action:

   - Make sure the decoder and the MTA are running (select STATUS from the X.400 Test Menu).
   - Make sure that decoder and MTA event logging are enabled (select STATUS from the X.400 Test Menu).
   - Check that the MTA routed the message to the correct oq number on the "Configure HP X.400/HP Desk Nodes" screen. If the delivery report was not routed to the correct oq number, then something was wrong with the originator's address in the original message or the routing to the HP X.400/HP Desk Node is not configured correctly.

9. NON Delivery Report arrived.

   Action:

   - Need the non-delivery report to see if the reason is given.
   - Check the "TRACKING OUTGOING MESSAGE" section of this test output.
     - If using X.400 with Sendmail, mailx or ELM, check if "X4MAILOUT event log" section of output indicates the message was sent to the MTA. If using X.400 with HP Desk, check if "encoder event log" section of output indicates the message was sent to the MTA.
     - Check if the "MTA event log" section of output indicates that the message was "RELAYED".

  - Check if the "RTS event log" section of output indicates that "OUTBOUND TRANSFER SUCCESSFULLY COMPLETED."

  If any of the above mentioned events are not logged, then it indicates that the outgoing message was not successfully sent. Proceed to the next item; otherwise, proceed to the item below that begins, "Check if the remote node...."

- Check if any other error message is reported by the test. If so, do the actions corresponding to that error message. Otherwise, proceed to the next item.
- Go through the events logged for the Outgoing message and get more information on each of the events from the *Installing and Configuring HP X.400 Administrator's Guide*.
- Check if the remote node generated a delivery report; if yes, find out the reason the address has been configured for this user and if the user exists.

If, after completing all of the above steps (1 through 9) you could not find or fix the problem, contact your HP representative with the information described under "Submitting Problem Information to HP" section in chapter 4 of this manual.

RTS IOP Testing

The RTS IOP tests in *osidiag* simulate RTS functionality and use the configuration information you supply when running the tests, *not* the information configured for X.400. Passing these tests tells you that you can connect to the remote RTS and that the NSAP, TSAP, SSAP, MTA name and password values you used to run the tests are correct. You must verify that the X.400 configuration is using these values.

If the previous tests passed (that is, if you were able to exchange messages with the remote system), you do not need to run the RTS Connection tests.

RTS IOP Testing with HP as Initiator

1. Log on as `root`.
2. Perform the Pre-Test Checklist steps.
3. Start *osiadmin* and select "X.400."
4. Select "Stop X.400."
5. Select "Test Connectivity."
6. Select "Status" and check that OTS is running, and RTS and MTA are stopped.

7. Select "Utilities" on the main menu.

8. Select "Tracing and Logging" on the Utilities menu.

   - Turn Tracing on for Transport, X.25 and LAN.

9. Select "Prompt Level" on the Utilities Menu, then select

   - "Prompt for Remote Address"
   - "Prompt for Local SAP"
   - "Prompt for Class"
   - "Prompt for other parameters"

10. Return to Previous menu.

11. Select "Simulate RTS Connect" on X.400 Test Cases menu.

12. Enter "SSAP," "TSAP," "NSAP" for the local system and press done. For X.400, the usual NULL SSAP and TSAP is "MHS." The SSAP, TSAP and NSAP can be entered in hex or ASCII. If you chose to enter data in ASCII text, you must enclose it with quotes to indicate so, for example:

    TSAP = 4d4853 or
    TSAP = "MHS"

13. Enter the SSAP, TSAP and NSAP for the remote system. For X.400, you usually have a NULL SSAP. The SSAP, TSAP and NSAP can be entered in hex or ASCII. If you chose to enter data in ASCII text you must enclose it with quotes to indicate so, for example:

    TSAP = 4d4853 or
    TSAP = "MHS"

Note The default value presented is that of the local RTS. You should overwrite this value.

---

14. Enter the X.400 Local MTA Name and the Outgoing Password configured for the adjacent MTA in *x4admin* (the X.400 MTA name and Password menu) and press "Perform Task" to run the test.

15. If the test passes, proceed to the section, "RTS Test with HP as Recipient." If the test fails, proceed to step 16.

16. Check the reason for failure returned by the test. The most common reasons and the actions to take are mentioned below:

    RTS Busy

    - Check the remote RTS status. If it is heavily loaded, try running the test later.
    - Check if the NSAP entered is correct. Try running the test again.

    Validation failure

    - Check the Password and MTA name configured and entered for the local and remote MTAs.
    - Check the NSAP configured and entered for the remote and local MTA.

    Unacceptable Dialogue mode

    - Check remote node for the dialogue mode configured for your MTA connection. HP only supports MONOLOGUE mode.

    If the test passes, proceed to the section, "RTS Test with HP as Recipient." If it fails, proceed to step 17.

    Another reason why this test can fail is mismatched Session functional units. The following functional units are supported by HP:

    - Kernel Functional Unit
    - Half Duplex Functional Unit
    - Minor Synchronize Functional Unit
    - Activity Management Functional Unit
    - Exceptions Functional Unit

17. Check the Session and Transport logs to see if more information on the cause of failure can be established.

18. If the problem persists, contact your Local HP Response Center.

RTS IOP Testing with HP as Recipient

If you are in *osiadmin*, X.400 Test cases and all steps to enable logging are unchanged from the previous test. Proceed to step 2.

1. Initial steps:
   - Log on as `root`.
   - Start *osiadmin* and select "X.400."
   - Select "Stop X.400."
   - Select "Test Connectivity."
   - Select Status and check that OTS is running, and RTS and MTA are stopped.
   - Select "Utilities" on the main menu.
   - Select "Tracing and Logging" on the Utilities menu.
     - Turn Tracing on for Transport, X.25 and LAN.
   - Select "Prompt Level" on the Utilities menu, then select
     - "Prompt for Remote Address"
     - "Prompt for Local SAP"
     - "Prompt for Class"
     - "Prompt for other parameters"
   - Return to Previous menu.
2. Select Utilities and increase "Wait Time" to at least 60 seconds.
3. Select "Simulate RTS Server" on X.400 Test Cases menu.
4. Enter "SSAP,""TSAP" for local system and press done. For X.400, the usual NULL SSAP and TSAP is "MHS." The SSAP and TSAP can be entered in hex or ASCII. If you chose to enter data in ASCII text, you must enclose it with quotation marks to indicate ASCII text, for example:

   TSAP = 4d4853 or
   TSAP = "MHS"
5. Enter the X.400 Local MTA Name and Outgoing Password configured for the adjacent MTA in the X.400 MTA name and Password menu and press "Perform Task" to run the test.

6. Go to the Remote System and send a message.

7. If the test passes, proceed to do the "User Agent and Message Transfer Agent Tests." If it fails, proceed to the next step.

8. Check the reason for failure returned by the test. The most common reasons and the actions to take are mentioned below.

   RTS Busy

   - Check the local RTS status. If it is heavily loaded, try running the test later.

   Validation failure

   - Check the Password and MTA name configured and entered for the local and remote MTAs.
   - Check the NSAP configured and entered for the remote and local MTA.

   Unacceptable Dialogue mode

   - Check remote node for the dialogue mode configured for your MTA connection. HP only supports MONOLOGUE mode.

   If the test passes, perform the "User Agent and Message Transfer" tests in the previous section. If it fails, proceed to the next step.

   Another reason why this test can fail is mismatched Session functional units. The following functional units are supported by HP:

   - Kernel Functional Unit
   - Half Duplex Functional Unit
   - Minor Synchronize Functional Unit
   - Activity Management Functional Unit
   - Exceptions Functional Unit

   If the test passes, perform the "User Agent and Message Transfer" tests in the previous section. If it fails, proceed to the next step.

9. Check the Session and Transport logs to see if more information on the cause of failure can be established.

10. If the problem persists, contact your Local HP Response center.

Testing MMS Interoperability

The steps below describe how to test MMS connectivity between the local and remote node. Chapter 7 contains more information about *osidiag*.

Pre-Test Checklist

1. **MMS Configured** - you should have configured at least one MMS application for the local node through *osiadmin*. Check the MMS Local Application worksheet.

Note *osidiag* will use the first configured application title which has "MMS" or "mms" as part of its title. (For example, the title `/O=hpindka/AP=MMS/AE=client1` could be chosen as a default, but `/C=US/O=foo/OU=bar/AP=yack/AE=ralf` would not.)

Hewlett-Packard recommends that you configure at least one MMS application following this rule.

2. **Local Stack Up** - Verify that the OTS stack has been started by running *otsstat*. If it is not up, run *osistart* to bring it up.
3. **Remote Stack Up** - Verify that the stack on the remote is up. If the remote is an HP system, then run *otsstat* on that system.
4. **Remote Presentation Address** - Determine the presentation address of the remote MMS application. This is made up of four components: the Presentation Selector (P-selector), the Session Selector (S-selector), the Transport Selector (T-selector) and the Network Address. See the Remote System Worksheet in the *Installing and Administering OSI Transport Services* manual for the remote presentation address.
5. **Remote MMS Process Up** - Verify that the remote has an MMS entity ready to accept a connection.
   - If the remote node is an HP system, you should run osidiag as a server on the remote node (see "Running the MMS Tests (Server Mode)" section) while running the test as a Client on the local node.
   - On non-HP equipment, the application you should run will vary from system to system. For Programmable Logic Controllers (PLCs) a server application may be started automatically. For other host systems, you should run a demo program or other MMS application capable of accepting a connection.

- If the remote node is only capable of initiating activity, then first prepare the local node as a server (see "Running the MMS Tests (Server Mode)" section) and then run the remote client application.

Running the MMS Tests (Client Mode)

Normally you will follow this list of steps to verify connectivity and interoperability with a remote node. If the remote is not capable of receiving connections, or you wish to test the remote's ability to establish connections, follow the instructions in "Running the MMS Tests (Server Mode)."

1. Log on as `root`.
2. Perform the "Pre-Test Checklist" if not already done.
3. Verify that logging is enabled by typing `/etc/nettl -status`. If the status display indicates that logging is not started, enable logging by typing `/etc/nettl -start`.
4. Run *osidiag* by typing `/etc/osidiag`. If the error "No Application Titles configured for MMS" appears on startup, then no applications were configured as described in step 1 of the pre-test checklist. Do one of the following:
   - Configure a local MMS application of this form.
   - Follow the steps under "Specifying Application Titles" later in this section.
5. Select "MMS Tests" from the menu.
6. Create a result file to keep a record of the test results by doing the following:
   - Select "Utilities" from the MMS menu.
   - Select "Open Result File" from the Utilities menu.
   - Enter the name of the file you wish to hold the results (for example, `/tmp/MMS.res`)
   - Press the "Done" key or [return].
   - You will see a message indicating that a result file was opened. Press the space bar.
   - Press the "Previous Menu" key to return to the MMS menu.

7. **STOP:** If there is not a server application running on the remote, then start one. For HP systems, starting a server is described in the following section, "Running the MMS Tests (Server Mode)." Proceed to step 8 after starting the server, or if one is already active, proceed to step 8.

8. Select "Connect..." from the MMS Test Case menu.

9. At this point, you should be asked for the MMS Server Presentation Address. Overwrite the default value presented with the address you got from step 4 of the Pre-Test checklist and press "Done."

   If you followed the additional steps under "Specifying an Application Title," you will be asked for a MMS Local Directory Distinguished Name and possibly a MMS Server Directory Distinguished Name. For these two screens, enter the DDNs you configured for a local MMS application and a remote MMS application respectively.

10. At this point, *osidiag* will attempt to form a connection with the remote.

11. Look at the reported test status near the bottom of the output. If it indicates PASSED, then you have successfully communicated with the remote node and you are finished with this section. If *osidiag* reports FAILED, look at the subsection Interpreting MMS Errors below.

    You should also check the status of the Server application on the remote node. For HP, this corresponds to step 11 in "Running the MMS Tests (Server Mode)."

    If you cannot find your error, or the recommended action is unsuccessful, then proceed with the next step.

12. Enable tracing by doing the following steps:

    - Select "Utilities" from the MMS menu.
    - Select "Tracing and Logging" from the Utilities menu.
    - Turn the logging on for MMS, ULA, and OTS. Turn tracing on for the Session Layer. You turn logging and tracing on by placing a Y after each item. Leave the other flags set to N.
    - Press the "Done" key (if you haven't already) to cause the change.
    - For the next two screens, use the default values by pressing "Done."
    - Press the "Previous Menu" key to return to MMS.

13. Rerun the connection test (steps 7-10). Note that any addressing information you entered in the previous steps is preserved by *osidiag*. You will not need to retype it.

14. Examine the tracing and logging information produced. See the subsection "MMS Logging Information" for more details. You may wish to examine the output with an editor, you can do this without terminating *osidiag* by doing the following:

    - Press return to get back to the MMS menu (if not already there).
    - Select "Utilities."
    - Select "Close Result File."
    - Press the "Shell" function key from the Utilities menu.
    - Use a text editor (for example, vi /tmp/MMS.res) to view the result file.
    - Exit the editor.
    - To return to *osidiag*, type exit.
    - Open a new result file (for example, `/tmp/MMS.res2`).
    - Return to the MMS menu.

At this point, you will be in one of the following situations:

- **MMS problem corrected** - If you have made a change suggested and verified that it worked, then you are done.
- **Configuration or other change required** - If the corrective action requires you to modify the configuration, then do so and rerun the verification steps. You should exit *osidiag* before rerunning the tests. You may also have to run the *update* program or reboot your system for configuration changes to take effect.
- **Apparent Lower Layer Problem** - If you have been unsuccessful in isolating the problem from the MMS layer, then proceed to the verification at the lower layers. Typically, this will be when you see a REJECT in the OTS log file.

Running the MMS Tests (Server Mode)

If the remote is not capable of receiving connections, or you wish to test the remote's ability to establish connections, follow these instructions.

Note The client application should simply initiate a connection and then disconnect for these steps to work.

---

1. Log on as `root`.
2. Perform the "Pre-Test Checklist" if not already done.
3. Verify that logging is enabled by typing `/etc/nettl -status`.
4. Run *osidiag* by typing `/etc/osidiag -w 300`. The -w option allows time (300 seconds) to get the client ready once the server is started.

   If the error "No Application Titles" configured for MMS appears on startup then no applications were configured as described in step 1 of the pre-test checklist. Do one of the following:

   - Configure a local MMS application of this form.
   - Follow steps under "Specifying Application Titles."
5. Select "MMS Tests" from the menu.
6. Create a result file to keep a record of the test results by doing the following:
   - Select "Utilities" from the MMS menu.
   - Select "Open Result File" from the Utilities menu.
   - Enter the name of the file you wish to hold the results (for example, `/tmp/MMS.res`).
   - Press the "Done" key or [return].
   - You will see a message indicating that a result file was opened, press the space bar.
   - Press the "Previous Menu" key to return to the MMS menu.
7. Select "Server..." from the MMS Test Case menu.

8. If the subsequent screen presented requests variable information, go to step 9. Otherwise, enter the address information as described in "Specifying Application Titles."

9. Leave the variable information screen unchanged, press "Done."

10. At this point, *osidiag* will listen for a connection from the remote. You should now generate the connection from the client system. If the other system is an HP, then you should follow the instructions in "Running the MMS Tests (Client Mode)" to do this. *osidiag* will display the progress of the test on the screen.

11. Look at the reported test status near the bottom of the output. If it indicates PASSED, then you have successfully communicated with the remote node and you are finished with this section. If *osidiag* reports FAILED, then look at the subsection "Interpreting MMS Errors"45 below. If you cannot find your error, or the recommended action is unsuccessful, then proceed with the next step.

12. Enable tracing on the local node as described in step 12 of the Client test.

13. Rerun the server test by selecting MMS Server from the menu. The values (if any) that you specified for the previous test are preserved so just press "Done" on any windows displayed.

14. Rerun the client test on the remote to connect to this server.

15. Examine the tracing and logging as described in step 14 of the client test. Follow the discussion after step 14 for further troubleshooting if necessary.

Specifying Application Titles

By default, the only addressing information *osidiag* will ask for on an MMS test is the remote Presentation address. This will be sufficient to run a test case except when *osidiag* cannot find a default local application title to use. (Remember that Application Titles are also referred to as Directory Distinguished Names (DDN).)

If you need to specify the local DDN, perform the following steps before running the test:

1. Select "Utilities" from the menu.
2. Select "Prompt Level" from the Utilities menu.
3. Put "Y" in front of the item labeled Local DDN.
4. Leave other fields unchanged and press "Done."
5. Press "Previous Menu."

When specifying the remote address, you have the choice of using a DDN or a presentation address. By default, *osidiag* only asks for a presentation address. If you wish to give a DDN, instead follow these steps before running the test:

1. Select "Utilities" from the menu.
2. Select "Prompt Level" from the Utilities menu.
3. Put a "Y" in front of the item labeled Remote DDN.
4. Leave other fields unchanged and press "Done."
5. Press "Previous Menu."

Interpreting MMS Errors

The following list describes possible errors and corrective actions. The list is sorted by the name of the function producing the error. The names are displayed by *osidiag* on the line immediately after the test status.

If you can't find your error here, then check the *HP MMS/9000 Reference Manual.*

mm_vactivation()

This routine is used to create a local VMD. It results in no end to end communication, nor does it use the services of the Service Provider Process. The common error for this operation is:

(MME194) Unknown ae_title

The address specified for the local DDN is not configured. Check the address shown at the top of the test case. It is on the line labeled "mm_my_ddn". Either rerun the test with a different local title, or configure the given title into OTS.

Note After making configuration changes, use *otsupdate* for them to take effect. See the *Installing and Administering OSI Transport Services* manual for more information.

See Pre-test checklist step 1 for more information about configuring.

mm_aeactivation()

This routine is used to create a local Service Provider Process. No end to end communication is performed by it. Errors with this routine may be caused by the following:

- OTS not running. You will typically see MME240 and MMV005 that indicate Connection management detected an error. Restart the stack.
- MAP still installed. Make sure you have switched stacks. Complete the OTS installation.

mm_connect()

This routine issues the MMS (and underlying layers) connection request and awaits a response (when running loopback, the em_wait() call is used to receive the response). The following are common errors during the connection:

- Bad presentation address. Any of the components of the address may be incorrect (for example, the P, S, T selector or the Network address). Check that the address you specified corresponds to that of the MMS application awaiting a connection. (NOTE: a presentation address is not necessary if you specify a Server Directory Name; however, if you specify both, the P-address overrides the name). Rerun with the correct address.
- Bad Server Directory Name. The value given for the MMS Server Directory Distinguished Name is not configured. This corresponds to an application title configured in the file, `/etc/net/osi/conf/remote_app`. The problem may be that the address you specified was not configured at all, or that the associated address was incorrect. Edit the remote_app file to either add an entry for the destination, or to correct the configured address, then run *otsupdate*. See *Installing and Administering OSI Transport Services* for more information on *otsupdate*.
- No MMS application waiting. There must be an application capable of receiving and responding to your connection request (unless running loopback). The server may be the *osidiag* test case MMS Server, an equivalent function from another vendor's equipment, or a PLC which is enabled. Start a server application running.
- MMS version incompatible. *osidiag* for OTS will by default propose the IS version of MMS. The remote may be running the DIS version. Check the remote's documentation to determine which it is. MAP 3.0 compliant systems run the DIS, not the IS. Set the prompt level for *osidiag* to ask for Class. Then, when you run the Connect test, you will be prompted for the version number for MMS, specify 0 rather than 1. This will cause DIS to be proposed.

em_wait()

This routine is called to wait for a scheduled MMS activity to complete. The most common error is a timeout when running the server. Set the time *osidiag* waits for a timeout. This can be done from the HP-UX system prompt (for example, `osidiag -w 120` to wait for 120 seconds) or from under the *osidiag* Utilities menu. Also verify that the remote application is actually sending the expected MMS operations to the local node.

MMS Logging Information

The previous steps may direct you to generate logging and tracing information. In general, logged errors will provide a more detailed indication of what went wrong than the error code returned through the API. The tracing information you gather may be useful if you have a good understanding of the OSI protocols; however, tracing information is primarily intended for use by the HP support personnel if you cannot solve the problem yourself.

Further information about the logging and tracing information is given in the following locations:

- Application Layer Logging - These are errors from the upper layers (in this case MMS and ULA). The error text should help you understand what went wrong. General information about reading the logged messages is given in chapter 6, "Logging and Tracing."
- OTS Logging - Common errors logged by OTS or the NETWORK, TRANSPORT or SESSION subsystems are described in chapter 4, "Troubleshooting." Detailed information about OTS logged messages is given in chapter 8, "Messages."
- Session Layer Tracing - A discussion of a Session Layer dialog is given at the end of the "Testing Session Interoperability" section of this chapter.

Testing FTAM Interoperability

The steps below describe how to test FTAM connectivity between the local and the remote node.

Pre-Test Checklist

You need to ensure the following conditions and get the following information before proceeding with the test:

1. **Local Stack Up** - Run *otsstat* to make sure the local stack is running over the appropriate link. Also ensure that the local FTAM Responder daemon is running. You can do this by issuing the command `ps -ef | fgrep ft` and see if a process named ftam_resp is active, or run *osistat.*
2. **Remote Stack Up** - Verify that the stack on the remote node and the FTAM service is running (for HP systems follow the procedure above for local stack up).
3. **Remote Presentation Address** - Determine the presentation address of the remote FTAM responder. This is made up of four components: the Presentation Selector ("P-selector"), the Session Selector ("S-selector), the Transport Selector ("T-selector") and the Network Address.

Running the FTAM Tests

Under ordinary circumstances, you will access FTAM through the *ftam(1)*, *fcp(1)*, *fmv(1)*, and *frm(1)* commands. However the procedure described here invokes FTAM through *osidiag.* The reason for this is to provide as much information as possible about errors that might occur and to create a result file that can be used by your HP support representative in the event you need support for diagnosing errors.

1. Log on as `root` on the local node. (Omit if already logged in.)
2. Perform the "Pre-Test Checklist" if not done already.
3. Verify that the FTAM service logging is enabled by typing `/etc/nettl -status`. If the status display indicates that logging is not started, enable logging by typing `/etc/nettl -start`.
4. Run *osidiag* by typing `/etc/osidiag`.
5. Select "FTAM Tests" from the main menu.

6. Create a result file which will keep a record of the test results by doing the following steps:

   a. Select "Utilities" from the FTAM menu.

   b. Select "Open Result File" from the Utilities menu.

   c. Enter the name of the file you wish to hold the results (for example, /tmp/ftam.res).

   d. Press the "Done" key (if you haven't already) to open the file.

   e. A message will be displayed indicating the file was opened. Press the space bar. If the message indicates some error occurred, check the name you specified.

   f. Press the "Previous Menu" key to return to the FTAM Menu.

7. Select "Connect" from the FTAM menu.

8. You will be prompted for the Initiator Identification. This is your login information for the remote node.

   The Initiator Identity field will contain your login name on the local system. You should change this unless the login you wish to use is the same on the remote.

   The Initiator Password is the password associated with the login on the remote node. When you type this it will not be displayed, so type carefully.

9. Press the "Done" key after entering this information.

10. The presentation address for the remote is requested next. Enter the value that you determined earlier.

    Note on entering values: *osidiag* will display the local FTAM Responder address by default. Overwrite this value.

11. The output from the test is passed through the HP-UX utility *more*. If the word "—More—" appears at the bottom of the screen, press the space bar to get the next screenful of information, or press return to get the next line.

12. Look at the reported "test status," near the bottom of the output. If it indicates PASSED, then you have successfully established an FTAM connection with the remote responder. Press [return] to return to the FTAM menu. Skip ahead to step 16 (labeled "File Transfer").

    If *osidiag* reports FAILED, then look at the subsection "Interpreting FTAM Errors" at the end of this list of steps. If you cannot find your error, or the recommended action is unsuccessful, proceed with the next step.

13. Enable tracing and logging by doing the following steps:

    a. Select "Utilities" from the FTAM menu.

    b. Select "Tracing and Logging" from the Utilities menu.

    c. Turn the logging on for the FTAM, ULA, and OTS layers. Turn tracing on for the Session Layer. You do this by placing the letter "Y" after each item. Leave the other fields set to "N."

    d. Press the "Done" key (if you haven't already) to cause the change in tracing to take effect.

    e. Press the "Previous Menu" key to return to the FTAM Menu.

14. Rerun the connection test by selecting "Connect" and pressing the "Done" key when prompted for the initiator information and again when prompted for the addresses. Note that *osidiag* will preserve the information you entered from the last run.

15. Examine the tracing and logging information produced. See the subsection ahead on "FTAM Logging Information" for more details. You may wish to temporarily leave *osidiag* to examine the output with an editor. To examine the output with an editor, do the following:

    a. Press [return] to get back to the FTAM menu.

    b. Select "Utilities."

    c. Select "Close Result File" from the Utilities menu.

    d. Press the "Shell" key to temporarily exit *osidiag*.

    e. Bring up an editor on the result file you created (for example, vi/tmp/ftam.res).

    f. Examine the logged information. See the subsection ahead on "FTAM Logging Information" for more details.

g. Exit the editor.

h. Return to *osidiag* by typing "exit"

i. Open a new result file by selecting Open Result File.

j. Specify a different name to avoid overwriting the old file (for example, /tmp/ftam.res2).

k. Press "Done," and then press the space bar.

l. Return to the FTAM menu by pressing the "Previous Menu" key.

At this point, you will be in one of the following situations:

- **FTAM connection problem corrected** - If you have made a change which corrected your FTAM problem then proceed to step 16 (labeled "File Transfer").

Note You may wish to turn tracing and logging off at this point. If so, repeat the steps outlined above for turning tracing on, except set all values to "N."

- **Configuration or other change required** - if the corrective action in the following sections require you to change a configuration parameter or make some other change, then do so and rerun the FTAM test. You may have to use the *otsupdate* program or reboot the system.
- **Apparent Lower Layer Problem** - If you have been unsuccessful in isolating the problem from the FTAM layer then proceed to the verification at the lower layers. Typically this will be when you have seen "REJECT" in the OTS log. If you are connected to the remote node via X.25, then proceed to the steps outlined in the "X.25" section. Otherwise proceed to the steps outlined in the "LAN" section.

16. **File Transfer.** At this point, it is assumed you have successfully connected to the remote node. You should be at the FTAM menu. If you previously closed the result file and have not opened a new one, do so now. Now select "Low Level Transfer."

17. You are now presented with two sets of initiator identification parameters. Leave the first initiator ID set to your local login name. Set the first password to your local password.

Leave the second set unchanged. These are the values that were successful on the connection test. Note that the password value is kept without being displayed. Press "Done."

18. Leave the Source Presentation Address unchanged. This is the address of the local FTAM responder. This indicates that the file will come from the local node and its destination will be the remote.

    Press "Done."

19. Leave the Destination Address unchanged. This is the address of the remote FTAM responder you entered previously, and it will receive the file to be transferred.

    Press "Done."

20. Assuming that your system has a "message of the day" configured, you can leave source and destination filenames unchanged. If you wish to transfer a different file, overwrite the source name. If you wish it to go to a different location, overwrite the destination. Press "Done."

21. Look at the reported "test status," near the bottom of the output. If it indicates PASSED, then you have successfully transferred a file to the remote node. FTAM Verification is now complete. Exit *osidiag* by pressing F8 twice. There is no need to perform lower layer verification.

    If *osidiag* reports FAILED, then look at the subsection "Interpreting FTAM Errors" at the end of this list of steps. If you cannot find your error, or the recommended action is unsuccessful proceed with the next step.

22. Re-enable tracing and logging if it had been turned off. Repeat the steps listed above with the heading "FTAM Tracing and Logging."

23. Rerun the test from step 1.

24. Refer to the section below on "FTAM Logging Information" for help in interpreting the output.

25. At this point you will be in one of the following situations:

    - **FTAM file transfer problem corrected** - FTAM has now been verified.
    - **Configuration change required** - If the corrective action in the following sections require you to change a configuration parameter, then do so and rerun the FTAM tests. You may have to run the *otsupdate* program or reboot the system.

- **Unsuccessful problem isolation** - If you succeeded in running the connection test, but did not succeed in performing the file transfer, the connection problem most likely lies in the local or remote FTAM entities. Go to the section titled "Submitting Problem Information to HP" in chapter 4.

Interpreting FTAM Errors

If present, the most helpful piece of error information is the field labeled "diagnostic." It begins four lines below the status FAILED. When present, the text string labelled "further details" will give you a good indication of the cause of the error.

Another useful piece of information is the operation which failed. The operation which failed is listed on the line after the status FAILED.

Below are the common FTAM calls which could fail and corrective actions which can be taken.

ft_aeactivation()

This command creates the "Service Provider Process" which acts as the FTAM initiator. Failures on this command may indicate the following:

- FTAM has not been correctly installed. Run *pdfck* on the FTAM fileset to verify all components are installed. If failure, reinstall from tape.
- The OTS stack is not up. Run the Status operation under Session or Transport to verify this. If not, start the stack.

ft_connect()

This operation attempts to connect to the remote node. Failures here may be indicative of the following:

- Incorrect address specified. If you specified a Presentation Address, then recheck the value you specified and the value configured for the remote.
- Incorrect user identification. You must provide valid user identification, user name and password to successfully connect. This typically corresponds to the remote system's login information.
- Remote stack not up. This should have been checked during the pre-test, but reverify at this point.

- Responder not running. The FTAM responder on the remote must be in a state where it can accept connections. Repeat the verification described in the pre-test section.
- Lower layer problem. There may be a problem at the lower layers of the stack. Go back to the step you were on and continue as directed.

ft_select()

This operation attempts to select a file to be read. This error typically occurs when the source file name you specified is incorrect.

ft_create()

This operation attempts to create a file on the remote. This error typically occurs if there are permission problems, or incorrect directories were specified in the path for the destination file name.

ft_sdata()

The low level transfer test currently can transfer files of at most 7000 bytes. If you attempt a larger transfer, you will receive an error code: 101 Buffer too large error. Rerun the test with a smaller file or use the High Level Transfer test.

ft_xxx()

Other FTAM operations not listed above are not common points of errors. One exception is if an abort indication is received. This may indicate that the remote went down for some reason. Check the remote stack and FTAM responder if an abort indication is indicated.

FTAM Logging Information

The steps above may direct you to generate logging and tracing information. In general, logged errors will provide a more detailed indication of what went wrong than the error code returned through the API. The tracing information you gather may be useful if you have a good understanding of the OSI protocols; however, tracing information is primarily intended for use by the HP support personnel if you cannot solve the problem yourself.

Further information about the logging and tracing information is given in the following locations:

- Application Layer Logging - These are errors from the upper layers (in this case FTAM and ULA). The error text should help you understand what went wrong. General information about reading the logged messages is given in chapter 6, "Logging and Tracing."
- OTS Logging - Common errors logged by OTS or the NETWORK, TRANSPORT or SESSION subsystems are described in chapter 4, "Troubleshooting." Detailed information about OTS logged messages is given in chapter 8, "Messages."
- Session Layer Tracing - A discussion of a Session Layer dialog is given at the end of the "Testing Session Interoperability" section of this chapter.

Testing APRI Interoperability

The steps below describe how to test the ACSE/Presentation and ROSE (APRI) layer connectivity between the local and remote node. This section is only recommended if you are developing APRI programs and wish to verify connectivity at that layer. In particular, this section is not recommended for diagnosing problems with the upper layer services (for example, FTAM, X.400, and MMS). See chapter 7 for detailed information about *osidiag*.

Pre-Test Checklist

1. **Local Stack Up** - Verify that the OTS stack has been started by running *otsstat*. If it is not up, type *otsstart* to bring it up.
2. **Remote Stack Up** - Verify that the stack on the remote is up. For an HP system you would use *otsstat*.
3. **Remote Presentation Address** - Determine the Presentation Address of the remote application. This is made up of four components: the Presentation Selector (P-selector), the Session Selector (S-selector), the Transport Selector (T-selector), and the Network Address.

   You should be able to gather this information from the "Remote System Worksheet" described in the *Installing and Administering OSI Transport Services* manual.
4. **Proposed Context List** - Determine the Presentation Contexts which the remote expects. *osidiag* will accept any Contexts, but by default will propose two contexts: ACSE (object ID {2 2 1 0 1}) and an arbitrary second value (object ID {2 1 9999 1}).
5. **ROSE Contexts** - Determine which contexts from step 4 will be used for ROSE. Skip this step if you are using only ACSE/Presentation.
6. **Application Context** - Determine the Application Context expected by the remote. *osidiag* will accept any application context, but by default will use the object ID {2 1 9999 2}.
7. **Remote ACSE/Presentation or ROSE Process Up** - Verify that the remote has a ACSE/Presentation entity ready to accept a connection.

   a. If the remote node is an HP system, you should run *osidiag* as a server on the remote node (see "Running the APRI Tests (Server Mode)" section) while running the test as a Client on the local node.

b. On non-HP equipment, the application you should run will vary from system to system. Typically, you should run a demo program or other ACSE/Presentation application capable of accepting a connection.

c. If the remote node is only capable of initiating activity, then first prepare the local node as a server, see "Running the APRI Tests (Server Mode)" section, and then run the remote client application.

8. **Determine Local NSAP** - By default, *osidiag* will bind to the CLNS over LAN NSAP if one is configured. If your configuration includes both CLNS and CONS subnetworks, then you will have to alter *osidiag*'s default behavior to communicate to remote nodes accessed through CONS. See "Changing Your Local NSAP" at the end of this section.

Running the APRI Tests (Client Mode)

Normally, you will follow this list of steps to verify connectivity and interoperability with a remote node. If the remote is not capable of receiving connections, or you wish to tests the remote's ability to establish connections, follow the instructions in "Running the APRI Tests (Server Mode)."

1. Log on as `root`.

2. Perform the "Pre-Test Checklist" if not already done.

3. Run *osidiag* by typing `/etc/osidiag`.

4. Select "ACSE/Presentation or ROSE Tests" from the menu.

5. Create a result file to keep a record of the test results by doing the following:

   a. Select "Utilities" from the Session menu.

   b. Select "Open Result File" from the Utilities menu.

   c. Enter the name of the file you wish to hold the results (for example, `/tmp/apri.res`).

   d. Press the "Done" key or [return].

   e. You will see a message indicating that a result file was opened. Press the space bar.

   f. Press the "Previous Menu" key to return to the Session menu.

6. **STOP:** If there is not a server application running on the remote, then start one. For HP systems, starting a server is described in the following section, "Running the APRI Tests (Server Mode)." Proceed to step 7 after starting the server, or if one is already active proceed to step 7.

7. Select "Connect..." from the Test Case menu.

8. You will be prompted for the destination Presentation Address. Enter the values recorded from step 3 of the Pre-Test Checklist.

Note On entering values: *osidiag* will display the local address by default. The default P, S and T selectors are given in ASCII surrounded by double quotes. If the address you must use is specified in hexadecimal rather than ASCII, then omit the double quotes (for example 22003176).

---

9. Next you will be prompted for the Proposed Contexts. Enter the values recorded from step 4 of the Pre-Test Checklist.

10. Next you will be prompted for the Application Context. Enter the values recorded from step 5 of the Pre-Test Checklist.

11. For ROSE only: Next you will be prompted for the contexts used for ROSE. Enter the context identifiers for the contexts.

12. At this point, *osidiag* will attempt to form a connection with the remote.

13. Look at the reported test status near the bottom of the output. If it indicates PASSED, then you have successfully communicated with the remote node and you are finished with this section.

    If *osidiag* reports FAILED, then look at the subsection Interpreting APRI Errors below. If you cannot find your error, or the recommended action is unsuccessful then proceed with the next step.

14. Enable tracing by doing the following steps:

    a. Select "Utilities" from the menu.

    b. Select "Tracing and Logging" from the Utilities menu.

    c. Turn the tracing on for the ACSE/Presentation layer and Session layer by placing a "Y" after them. Also turn on OTS logging. Leave the other flags set to "N."

d. When you have finished selecting tracing options, press the "Done" key to implement the change.

e. You will then be prompted with two more windows for the trace and log levels. Use the default values by pressing the "Done" key for each.

f. Press the "Previous Menu" key to return to APRI.

15. Rerun the connection test (steps 6-13). Note that the information you entered in the previous steps is preserved by *osidiag*, so you will not need to retype it.

16. Examine the tracing and logging information produced. See the subsection ahead on "ACSE/Presentation Tracing and Logging Information" for more details. You may wish to examine the output with an editor. You can do this without terminating *osidiag* by doing the following:

    a. Press return to get back to the menu (if not already there).

    b. Select "Utilities."

    c. Select "Close Result File."

    d. Press the "Shell" function key from the Utilities menu.

    e. Use a text editor (for example, `vi /tmp/sess.res`) to view the result file.

    f. Exit the editor.

    g. To return to *osidiag*, type `exit`.

    h. Open a new result file (for example, `/tmp/apri.res2`)

    i. Return to the menu.

17. At this point, you will be in one of the following situations:

    - ACSE/Presentation problem corrected - If you have made a change suggested and verified that it worked then you are done.
    - Configuration or other change required - If the corrective action requires you to modify the configuration, then do so and rerun the verification steps. You should exit *osidiag* before rerunning the tests.
    - Apparent Lower Layer Problem - If you have been unsuccessful in isolating the problem from the APRI tests, then proceed to the verification at the lower layers.

Typically this will be when you see a REJECT in the OTS log. If you are connected to the remote via X.25 then proceed to the steps outlined in the X.25 interoperability procedures section. Otherwise proceed to the steps outlined in the LAN interoperability procedures section.

Running the APRI Tests (Server Mode)

If the remote is not capable of receiving connections, or you wish to tests the remote's ability to establish connections, follow these instructions.

Note The remote application should simply initiate a connection and then disconnect for these steps to work. Do not let the remote issue the connect request until you have gotten to step 9 of this list. If you attempt to issue the connect too early, it will not be received and the remote will report an error.

---

1. Log on as `root`.
2. Perform the Pre-Test Checklist if not already done.
3. Run *osidiag* by typing `/etc/osidiag -w 300`. The `-w` option allows time (300 seconds) to get the client ready once the server is started.
4. Select "ACSE/Presentation or ROSE Tests" from the menu.
5. Create a result file to keep a record of the test results by doing the following:
   a. Select "Utilities" from the ACSE/Presentation menu.
   b. Select "Open Result File" from the Utilities menu.
   c. Enter the name of the file you wish to hold the results (for example, `/tmp/apri.res`).
   d. Press the "Done" key or [return].
   e. You will see a message indicating that a result file was opened. Press the space bar.
   f. Press the "Previous Menu" key to return to the menu.
6. Select "Server..." from the Test Case menu.

7. For ROSE only: You will be asked for the Presentation context identifiers to be used as ROSE contexts. Enter the contexts determined in step 5 of the Pre-Test Checklist.

8. For ROSE only: You will be asked if you want *osidiag* to autorespond to ROSE invocations. Leave this set to "Y."

9. At this point, *osidiag* will listen for a connection from the remote.

   You should now generate the connection from the client side via the remote application. If the other node is an HP system then you should follow the steps in this chapter called "Running the APRI Tests (Client Mode)" to do this.

10. *osidiag* will display the progress of the test on the screen.

11. Look at the reported test status near the bottom of the output. If it indicates PASSED, then you have successfully communicated with the remote node and you are finished with this section.

    If *osidiag* reports FAILED, then look at the subsection "Interpreting APRI Errors" below. If you cannot find your error, or the recommended action is unsuccessful, then proceed with the next step.

12. Enable tracing on the local node as described in step 14 of the Client test.

13. Rerun the server test (steps 6 - 8).

14. Rerun the client test on the remote to connect to this server.

15. Examine the tracing and logging as described in step 13 of the client test. Follow the discussion after step 13 for further troubleshooting if necessary.

Changing Your Local NSAP

When APRI binds to a local address via ap_set_env(AP_BIND_PADDR), the value used for the NSAP will determine which remote systems you can communicate with. Specifically, if the NSAP given corresponds to a CLNS subnetwork (either over LAN or X.25), you will be able to communicate with all remote nodes accessible through CLNS. If the NSAP corresponds to a CONS subnetwork, then you will be able to communicate with remotes accessible through CONS.

Because *osidiag* uses a default value for your local NSAP, you may need to override the default in order to communicate with a desired remote. There are two ways you can modify the local NSAP used. The easier way is to run a loopback test over the desired NSAP. Because *osidiag* preserves values across tests in the same *osidiag* invocation, your new local NSAP will be used on subsequent tests to the remote.

The alternative mechanism is as follows:

1. Select "Utilities" from the menu.
2. Select "Prompt Level" from the Utilities menu.
3. Change the value for "Prompt for Local SAP" to "Y". Leave all other values unchanged. Then press "Done."
4. Press "Previous Menu."

After following these steps you will be prompted for the local as well as the remote address when you run a connection or server test.

Interpreting APRI Errors

The following list describes possible errors and corrective actions. The list is sorted by the name of the function producing the error. The names are displayed by *osidiag* on the line immediately after the test status.

If you can't find your error here, then check the *ACSE/Presentation and ROSE Interface (APRI) Programmer's Guide*. You may also want to look at chapter 7 of this manual for more information about *osidiag*.

ap_open()

This command is used to open an APRI endpoint. It is the first APRI command issued and results in local communication with the OTS stack. Failure is typically the result of the stack not being up. Another possibility is incorrect installation of the product. Run *otsstat* to see if the stack is indeed up.

ap_set_env()

This command is used to set APRI environment variables which determine the characteristics of this endpoint.

The most common problem here is that the local address (parameter ap_my_psap) is invalid. This would usually be a result of your changing the default value. Examine the local address used for the test.

ap_poll()

This command is used to receive indications and confirmations off the network. The two common errors encountered by this command are:

- time out - By default, *osidiag* will wait only 30 seconds for an indication or for a confirmation. After this time expires the operation is considered failed. A time out may indicate that the remote is not sending any response to your request. In the case of a server it may indicate that the remote is not initiating any activity. Verify that the remote is indeed performing its end of the dialog. If the timeout is too short, it may be changed under the Utilities menu.
- unanticipated primitive - *osidiag* received an indication which it did not expect. The name of the indication received will be displayed immediately after the call to ap_poll().

ap_rcv(A_PABORT_IND)

This operation is called to decode an incoming provider abort indication. The information carried on the abort is displayed, the source, reason and event causing the abort. More details about the meaning of these codes is provided in the APRI manual pages describing A_PABORT_IND.

The most common cause of this type of error is an incorrect remote address. Also make sure your local NSAP is on the same subnetwork as the destination system. See Pre-Test Checklist item 7.

ap_rcv(A_ABORT_IND)

This operation is called to decode an incoming user abort indication. Any data carried on the abort is shown. Because a user abort indicates that the application on top of the Presentation layer detected some problem, you should examine the output of the remote application for further information as to why the abort was sent. An abort may also be sent by the HP provider if the specified address is valid, but no process is currently accepting connections.

ap_rcv(A_ASSOC_CNF)

This operation is called to decode an inbound connection confirmation. The confirmation carries on a status value which indicates whether the connection was accepted or refused. A refusal indicates that your connect request arrived at the remote, but that the remote did not like one of your proposed values or it is not available to service connections. The confirmation carries three pieces of information: the result, the source (if rejected), and a diagnostic code.

More details about the meaning of these codes is provided in the APRI manual pages describing A_ASSOC_CNF.

ro_bind ( )

This operation associates the contexts IDs you specified in step 11 of the Client test with ROSE. Failure typically indicates that the values you specified are not compatible with those negotiated. Verify that the values for *ap_p_ctx_list* and *rose_pci_list* are consistent.

APRI Tracing and Logging Information

The steps above may direct you to generate logging and tracing information. In general, logged errors will provide a more detailed indication of what went wrong than the error code returned through the API. The tracing information you gather may be useful if you have a good understanding of the OSI protocols; however, tracing information is primarily intended for use by the HP support personnel if you cannot solve the problem yourself.

Further information about the logging and tracing information is given in the following locations:

- OTS Logging - Common errors logged by OTS or the NETWORK, TRANSPORT or SESSION subsystems are described in chapter 4, "Troubleshooting." Detailed information about OTS logged messages is given in chapter 8, "Messages."
- Session Layer Tracing - A discussion of a Session Layer dialog is given at the end of the "Testing Session Interoperability" section of this chapter.

Testing Session Interoperability

The steps below describe how to test Session layer connectivity between the local and remote node. This section is not recommended if you are troubleshooting a problem with the upper layer services (for example, FTAM, X.400, and MMS). For checking mid-stack connectivity with these applications, you should perform the steps described in the OTS section of this chapter. See chapter 7 for more information about *osidiag*.

Pre-Test Checklist:

1. **Local Stack Up** - Verify that the OTS stack has been started by running *otsstat*. If it is not up, type `otsstart` to bring it up.
2. **Remote Stack Up** - Verify that the stack on the remote is up. For an HP system you would use *otsstat*.
3. **Remote Session Address** - Determine the session address of the remote Session application. This is made up of three components: the Session Selector (S-selector), the Transport Selector (T-selector) and the Network Address.
4. **Remote Session Process Up** - Verify that the remote has a Session entity ready to accept a connection.
   a. If the remote node is an HP system, you should run *osidiag* as a server on the remote node (see "Running the Session Tests (Server Mode)" section) while running the test as a Client on the local node.
   b. On non-HP equipment, the application you should run will vary from system to system. Typically you should run a demo program or other Session application capable of accepting a connection.
   c. If the remote node is only capable of initiating activity, then first prepare the local node as a server (see "Running the Session Tests (Server Mode)" section) and then run the remote client application.

Running the Session Tests (Client Mode)

Normally you will follow this list of steps to verify connectivity and interoperability with a remote node. If the remote is not capable of receiving connections, or you wish to tests the remote's ability to establish connections, follow the instructions in "Running the Session Tests (Server Mode)."

1. Log on as `root`.
2. Perform the "Pre-Test Checklist" if not already done.
3. Run *osidiag* by typing `/etc/osidiag`.
4. Select "Session Tests" from the menu.
5. Create a result file to keep a record of the test results by doing the following:

   a. Select "Utilities" from the Session menu.

   b. Select "Open Result File" from the Utilities menu.

   c. Enter the name of the file you wish to hold the results (for example, `/tmp/sess.res`).

   d. Press the "Done" key or [return].

   e. You will see a message indicating that a result file was opened. Press the space bar.

   f. Press the "Previous Menu" key to return to the Session menu.
6. **STOP**: If there is not a server application running on the remote, then start one. For HP systems, starting a server is described in the following section, "Running the Session Tests (Server Mode)." Proceed to step 7 after starting the server, or if one is already active, proceed to step 7.
7. Select "Connect..." from the Session Test Case menu.
8. You will be prompted for the destination Session Address. Enter the values recorded from step 3 of the Pre-Test Checklist.

Note On entering values: *osidiag* will display the local address by default. The default S and T selectors are given in ASCII surrounded by double quotes. If the address you must use is specified in hexadecimal rather than ASCII, then omit the double quotes (for example, 22003176).

---

9. At this point *osidiag* will attempt to form a connection with the remote.

10. Look at the reported test status near the bottom of the output. If it indicates PASSED, then you have successfully communicated with the remote node and you are finished with this section.

    If *osidiag* reports FAILED, then look at the subsection "Interpreting Session Errors" below. If you cannot find your error, or the recommended action is unsuccessful, then proceed with the next step.

11. Enable tracing by doing the following steps:

    a. Select "Utilities" from the Session menu.

    b. Select "Tracing and Logging" from the Utilities menu.

    c. Turn OTS logging and tracing on for the Session Layer and Transport layer by placing a "Y" after them. Leave the other flags set to "N."

    d. For the next two screens, press the "Done" key to accept the default values.

    e. When you have finished selecting tracing options, press the "Done" key to implement the change.

    f. Press the "Previous Menu" key to return to Session.

12. Rerun the connection test (steps 6 through 9). Note that any addressing information you entered in the previous steps is preserved by *osidiag*, so you will not need to retype it.

13. Examine the tracing and logging information produced. See the subsection ahead on "Session Tracing and Logging Information" for more details. You may wish to examine the output with an editor. You can do this without terminating *osidiag* by doing the following:

    a. Press [return] to get back to the Session menu (if not already there).

    b. Select "Utilities."

    c. Select "Close Result File."

    d. Press the "Shell" function key from the Utilities menu.

    e. Use a text editor to view the result file (for example, `vi /tmp/sess.res`).

    f. Exit the editor.

g. To return to *osidiag*, type `exit`.

h. Open a new result file (for example, `/tmp/sess.res2`).

i. Return to the Session menu.

At this point, you will be in one of the following situations:

- **Session problem corrected** - If you have made a change suggested and verified that it worked, then you are done.
- **Configuration or other change required** - If the corrective action requires you to modify the configuration, then do so and rerun the verification steps. You should exit *osidiag* before rerunning the tests.
- **Apparent Lower Layer Problem** - If you have been unsuccessful in isolating the problem from the Session layer, then proceed to the verification at the lower layers. Typically, this will be when you see a T_REJECT in the OTS trace. If you are connected to the remote via X.25, then proceed to the steps outlined in the X.25 interoperability procedures section. Otherwise proceed to the steps outlined in the LAN interoperability procedures section.

Running the Session Tests (Server Mode)

If the remote is not capable of receiving connections, or you wish to test the remote's ability to establish connections, follow these instructions.

Note The remote application should simply initiate a connection and then disconnect for these steps to work. Do not let the remote issue the connect request until you have gotten to step 7 of this list. If you attempt to issue the connect too early it will not be received and the remote will report an error.

---

1. Log on as `root`.
2. Perform the Pre-Test Checklist if not already done.
3. Run *osidiag* by typing `/etc/osidiag -w 300`. The `-w` option allows time (300 seconds) to get the client ready once the server has been started.
4. Select "Session Tests" from the menu.

5. Create a result file to keep a record of the test results by doing the following:

   a. Select "Utilities" from the Session menu.

   b. Select "Open Result File" from the Utilities menu.

   c. Enter the name of the file you wish to hold the results (for example, `/tmp/sess.res`)

   d. Press the "Done" key or [return].

   e. You will see a message indicating that a result file was opened. Press the space bar.

   f. Press the "Previous Menu" key to return to the Session menu.

6. Select "Server..." from the Session Test Case menu.

7. At this point, *osidiag* will listen for a connection from the remote.

   You should now generate the connection from the client side via the remote application. If the other node is an HP system, then you should follow the steps in this chapter called "Running the Session Tests (Client Mode)" to do this.

8. *osidiag* will display the progress of the test on the screen.

9. Look at the reported test status near the bottom of the output. If it indicates PASSED, then you have successfully communicated with the remote node and you are finished with this section.

   If *osidiag* reports FAILED, then look at the subsection "Interpreting Session Errors" below. If you cannot find your error, or the recommended action is unsuccessful, then proceed with the next step.

10. Enable tracing on the local node as described in step 11 of the Client test.

11. Rerun the server test (steps 6 - 8).

12. Rerun the client test on the remote to connect to this server.

13. Examine the tracing and logging as described in step 13 of the client test. Follow the discussion after step 13 for further troubleshooting if necessary.

Interpreting Session Errors

The following list describes possible errors and corrective actions. The list is sorted by the name of the function producing the error. The names are displayed by `osidiag` on the line immediately after the test status.

If you can't find your error here, then check the *Session Access Programmer's Guide*. You may also want to look at chapter 7 of this manual for more information about *osidiag*.

osi_init()

This command is used to initialize the session library in preparation for communication with OTS. Failure is most often caused by a lack of available swap space.

osi_rgr_rq(), osi_rgr_cf()

These commands are used to register with the OTS stack. Possible errors are that the stack is not up, or that another application is already listening on this address or has requested exclusive access to this address.

osi_get_event()

This command is used to receive indications and confirmations off the network. The two common errors encountered by this command are:

- **time out** - By default *osidiag* will wait only 30 seconds for an indication or for a confirmation. After this time expires the operation is considered failed. A time out may indicate that the remote is not sending any response to your request. In the case of a server it may indicate that the remote is not initiating any activity. Verify that the remote is indeed performing its end of the dialog. If the timeout is too short it may be changed under the Utilities menu.
- **unanticipated primitive** - *osidiag* received an indication which it did not expect. The name of the indication received will be displayed immediately after the call to osi_get_event().

ses_pabort_id()

This operation is called to decode an incoming provider abort indication. The reason codes and their likely meaning are described in the "Protocol Reason Codes" section of chapter 8. See the description on "Interpreting Transport Errors" in the OTS interoperability procedures section of this chapter.

The abort reason code is found in the middle of the *osidiag* output and looks like this:

```
ses_pab_id()          pipe_fd:        0x00000008  result:  0
           abort reason code:        F3
                     meaning:        Invalid address or
                                     permanent Transport error
```

ses_uabort_id()

This operation is called to decode an incoming user abort indication. Any data carried on the abort is shown. Because a user abort indicates that the application on top of the Session layer detected some problem, you should examine the output of the remote application for further information as to why the abort was sent.

ses_connect_rf()

This operation is called to decode a refusal to your connection request. A refusal indicates that your connect request arrived at the remote, but that the remote did not like one of your proposed values or it is not available to service connections.

The refuse carries a code on it that may provide more information. This code is displayed in the in the middle part of the *osidiag* output, it looks like this:

```
ses_ref_cf()          conn_fd:        0x00000001  result:  0
                refuse code:        82
                     meaning:        No user at S-Selector
```

See possible disconnect codes and suggested actions in the "Interpreting Transport Errors" portion of the OTS interoperability procedures section of this manual.

OTS Logging Information

When run under *osidiag*, any logged errors will be displayed first, followed by any stack trace output. OTS logging will display problems detected by the stack in communicating with the remote. Chapter 8, "Messages," contains many examples and possible causes of these logged errors. Chapter 4, "Troubleshooting," describes some common logged errors and their causes.

Session Layer Tracing

Session trace records have the following format:

```
^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^HP OSI MGMT^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^

  Timestamp            : Wed Dec 17 1991 13:03:06.066190
  Process ID           : ICS:                  Subsystem        : SESSION
  User ID ( UID )      : -1                    Trace Kind       : HDR IN TRACE
  Device ID            : -1                    Path ID          : -1
  Connection ID        : -1

[9973] genmai2.c 1643
[....] S-CONNECT-Ind Buffer1 is Ses Id Buffer2 is Udta Sap=0000000b Size=
[....] 00000ff4 Ini Ser Num=00000000 Ini tk pos=00 Srv=00 FU=c249 SSAP Add=
[....] 0473657364 ..=0473657364 ..=076870696e646b61 Capa=0100
```

Trace entries will typically be a header in/out that describes the type of primitive, or a PDU in/out that describes any data carried by the preceding logged message.

More information about tracing can be found in chapter 6, "Logging and Tracing," and in chapter 8, "Messages."

For a normal Session dialog, the sequence of headers you should expect is as follows.

- **S-CONNECT-Req** - This is a request to the Session layer to connect to the remote.
- **S-CONNECT-Cnf** - This entry combined with any "PDU IN TRACE" entries display information about the connect confirm PDU we received.
- **S-DATA** - This entry depending on its suffix is either outbound (Req) or inbound (Ind) Session data. For the connect test no data will be present. Interpreting the contents of the data displayed requires knowledge of the Presentation and upper layer encoding rules (described in the appropriate ISO specifications).
- **S-READY** - Precedes the arrival and transmission of some PDUs.
- **S-P-ABORT-Ind (error)** - This indicates that the Session provider is issuing an abort. See the Cause field for more information. It has the same format as for T_REJECT described in the "Protocol Reason Codes" section in chapter 8.

- **S-U-ABORT-Ind (error)** - This indicates that the remote Session user abruptly released the connection. This can occur if the remote user of the Session service (for example, FTAM, MMS, Session application) aborts, or detects an unrecoverable error.
- **S-U-ABORT-Req (error)** - This indicates that the local Session user (in this case *osidiag*) requested that the Session connection be aborted. Check the *osidiag* output for information as to why *osidiag* aborted the connection. Typically this will be due to a timeout or the receipt of an unexpected indication.
- **S-CONNECT-Conf(Negative) (error)** - This entry indicates that the connect at the Session layer was received, but was rejected for some reason. The most common reasons are incorrect Session or Presentation Selectors. The Reason codes defined by ISO are as follows:

  00 - Reason not specified. Check the S- and P-selectors. Also check any logged errors on the remote.

  01 - Rejection by Session due to congestion. Verify the selectors, and that the user of the Session service (for example, FTAM, MMS, Session application) is up.

  02 - Rejection by called SS-user. If there is user data on the reject, it may contain Presentation layer data. Your S-selector is probably correct, but you may have an incorrect P-selector. The problem may also be an error detected by the remote user of the Session service (for example, FTAM, MMS, or Session application). Check the login information you supplied.

  81 - SSAP identifier unknown. Check the S-selector you specified.

  82 - SS-user not attached to SAP. Verify that the remote user of the Session service (for example, FTAM, MMS, Session application) is still running.

  83 - Session Protocol Machine congestion at connect time. This is also an unlikely error. Verify the selectors and that the user of the Session service (for example, FTAM, MMS, Session application) is up.

  84 - Proposed protocol versions not supported. The Session layer proposes versions 1 and 2. You should not get this error. Contact the support representative from the Remote system vendor.
- **S-RELEASE-Req** - This entry is created when we request a normal disconnect.
- **S-RELEASE-Cnf** - This entry is created when we receive the disconnect confirmation from the remote.

Testing OTS Interoperability

The steps below describe how to test Transport layer connectivity between the local and the remote node.

Pre-Test Checklist

You need to ensure the following conditions and get the following information before proceeding with the test:

1. **Local Stack Up** - Run *otsstat* to make sure the local stack is running over the appropriate link. If OTS is not running, type `otsstart` to start the stack. You should only use *otsstart* after initial configuration.

2. **Remote Stack Up** - Verify that the stack on the remote node is running. This includes the link products (LAN, X.25), OTS, and any services you may be running, such as X.400, MMS, and FTAM. On HP systems, the commands to verify the status are *otsstat* for the stack, /usr/lib/x400/x4stat for X.400 and *osistat* for FTAM Responder.

3. **Remote Network Address and Transport Selector** - Determine the network address of the remote node.

   If you have configuration worksheets for the FTAM Remote Responder or for Configuring Adjacent MTA, use the values on these.

   Alternatively for HP systems, these addresses can be determined as described in the X.400, MMS and FTAM Pre-test checklists.

4. **Determine Local NSAP** - By default *osidiag* will bind to the CLNS over LAN NSAP if one is configured. If your configuration includes both CLNS and CONS subnetworks, then you will have to alter *osidiag*'s default behavior to communicate to remote nodes accessed through CONS. See "Changing Your Local NSAP" at the end of this section.

Running the OTS Tests

1. Log on as `root` on the local node. (Omit if already logged on).
2. Perform the "Pre-Test Checklist" if not done already.
3. Run *osidiag* by typing `/etc/osidiag.`
4. Select "Transport Tests" from the main menu.
5. Create a result file which will keep a record of the test results by doing the following steps:
   a. Select "Utilities" from the Transport Tests menu.
   b. Select "Open Result File" from the Utilities menu.
   c. Enter the name of the file you wish to hold the results (for example, /tmp/tran.res).
   d. Press the [return] key (if you haven't already) to open the file. A message indicating the file was opened will be displayed. Press the space bar. If the message indicates some error occurred, check the name you specified. Press the "Previous Menu" key to return to the Transport Tests Menu.
6. Select "Connect" from the Transport Tests menu.
7. You will be prompted for the destination Transport Selector and Network Address. Enter the values you recorded earlier. Use the worksheet values if available. Press the "Done" function key when values are entered.

Note *osidiag* will display the local addresses by default. The default T-Selector is specified in ASCII, surrounded by double quotes. If the selector and/or address you have is specified in hexadecimal rather than ASCII, then omit the quotation marks (for example, 220011).

---

8. Look at the reported test status, near the bottom of the output. If it indicates PASSED, you've successfully verified OTS interoperability with a remote system.

   If you've performed remote interoperability procedures for X.400, FTAM, or MMS which failed, the problems are at the Session layer or above. Re-examine the results of your Application layer run which you stored in a result file (for example, /tmp/ftam.res). See if the errors you saw make any more sense with the knowledge that the Transport test passed. If not, then go to the "Submitting Problem Information to HP" section in chapter 4.

Changing Your Local NSAP

When XTI binds to a local address via t_bind(), the value used for the NSAP will determine which remote systems you can communicate with. Specifically if the NSAP given corresponds to a CLNS subnetwork (either over LAN or X.25), you will be able to communicate with all remote nodes accessible through CLNS. If the NSAP corresponds to a CONS subnetwork, then you will be able to communicate with remotes accessible through CONS.

Because *osidiag* uses a default value for your local NSAP, you may need to override the default in order to communicate with a desired remote. There are two ways you can modify the local NSAP used. The easier way is to run a loopback test over the desired NSAP. Because *osidiag* preserves values across tests in the same *osidiag* invocation, your new local NSAP will be used on subsequent tests to the remote.

The second way to modify the local NSAP is as follows:

1. Select "Utilities" from the menu.
2. Select "Prompt Level" from the Utilities menu.
3. Change the value for "Prompt for Local SAP" to "Y." Leave all other values unchanged. Then press "Done."
4. Press "Previous Menu."

After following these steps you will be prompted for the local as well as the remote address when you run a connection or server test.

If the OTS Test Failed

If *osidiag* reports FAILED, then look at the subsection "Interpreting Transport Errors" at the end of this list of steps. If you cannot find your error, or the recommended action is unsuccessful, proceed with these steps.

1. Enable logging and tracing by doing the following steps:
   a. Select "Utilities" from the Transport menu.
   b. Select "Tracing and Logging" from the Utilities menu.

   c. Turn on logging for OTS and tracing on for Transport Layer and Network Layer by placing the letter "Y" after them. Leave the other fields set to "N."
   d. Press the "Done" key (if you haven't already) to cause the change in tracing to take effect. Also, press "Done" after the next two screens to accept the default log and trace values.
   e. Press the "Previous Menu" key to return to the Transport Menu.

2. Rerun the connection test by selecting "Connect" and pressing the "Done" key when prompted for the addresses. Note that *osidiag* will preserve the addresses you entered from the last run.
3. Examine the tracing and logging information produced. See the subsection ahead on logging information for more details. You may wish to temporarily leave *osidiag* to examine the output with an editor. To examine the output with an editor, do the following:
   a. Select "Utilities."
   b. Select "Close Result File" from the Utilities menu.
   c. Press the "Shell" key to temporarily exit *osidiag*.
   d. Bring up an editor on the result file you created (for example, vi /tmp/tran.res).
   e. Examine the logged information. See the section on "Common Logged Errors" at the end of chapter 4 for more details.
   f. Exit the editor.
   g. Return to *osidiag* by typing "exit."
   h. Return to the Transport menu by pressing the "Previous Menu" key.
4. At this point you will be in one of the following situations:
   - **Transport problem corrected** - If you have made a change which corrected your Transport problem then return to the Service layer test which originally failed and try again.
   - **Configuration or other change required** - If the corrective action in the following sections require you to change a configuration parameter or make some other change, then do so and rerun the Transport test.
   - **Problem persists** - If you have been unsuccessful in correcting the problem, gather the information you have collected to provide to your HP support

representative. See the section "Submitting Problem Information to HP" in chapter 4.

Interpreting Transport Errors

If the transport test fails, *osidiag* will report which operation encountered the error. The common operations and likely interpretations are listed below.

t_open()

If we fail to open a Transport endpoint, it indicates a local problem. Because it is the first function called by *osidiag* for the transport tests, you may see this if you have not brought up the stack. Verify by running the status operation under the Transport menu.

t_bind()

If we fail to associate an address with the transport endpoint, this indicates a likely parameter error. Verify that the parameter "tp-my-tsap" is set to "diagt AP" where AP is a valid local address. This error may also occur if the link supporting this address went down. Run *otsstat.*

t_rcvdis()

This operation is called to decode a disconnect indication. When a disconnect is received, any data carried is displayed along with the disconnect code. The disconnect code is shown in hex and the meaning of this code is displayed.

Possible disconnect codes and suggested actions are given in chapter 8, "Messages," in the section "Protocol Reason Codes."

t_look()

This operation waits for an event to arrive from the remote. It will wait for a default time of 30 seconds. Failure to receive a response in this time typically indicates that the remote stack is not enabled, or is failing to respond.

One possible reason that a remote stack would not respond is if one node is configured as NULL internet and the other is not.

Another possibility with NULL internet is that the remote network address is incorrect.

OTS Logging

The log output is displayed immediately after *osidiag*'s test status report. See the discussion of "Common Logged Errors" at the end of chapter 4, "Troubleshooting." Also see chapter 8, "Messages."

Transport and Network Layer Tracing

Transport layer tracing is performed by *nettl*. See chapter 6 for more details about *nettl*. The transport dialog sequence you should expect is as follows.

- **T-Connect-Req** - This indicates that the connection request has been received from *osidiag* and is ready to be sent. If this record is not present, you have either not enabled Transport Tracing, or *osidiag* is not able to access the stack (recheck errors reported).
- **N-UNIT-DT-Request** - This entry indicates that the a PDU was sent out on the network. If there is not one of these entries between a T-Connect-Req and a T_REJECT entry, it indicates that the connection failed locally, probably unknown address.
- **N-UNIT-DT-Indication** - You may see several of these entries apparently unrelated to your Transport dialog. This may be because you are using the ES/IS protocol on your network. If the data buffer after the record has a value of 0x82 as its first byte then it is an ES/IS PDU. A value of 0x81 indicates that it is CLNS protocol.
- **N-CONNECT-Req** - This trace message indicates a Network connection is being established over X.25. You may not see this message traced if your connection is being multiplexed over an existing X.25 connection, or reusing a connection which was released a short time ago.
- **N-DATA-Req** - This entry shows data being carried over a CONS network connection. The data carried will be a Transport Layer PDU.
  - **T-CONNECT-Cnf** - This entry combined with the T_ESTAB above show us that the remote did accept your connection.
  - **T-DATA** - This entry depending on its suffix is either outbound (Req) or inbound (Ind) Transport data. For the connect test no data will be present.
  - **T-DISCONNECT-Req** - This entry is created when we request a normal disconnect.

Testing LAN (802.3 or FDDI) Interoperability

The steps below describe how to test the connectivity at the Link level (either 802.3 or FDDI LAN) between the local and the remote node.

Pre-Test Checklist

You need to ensure the following conditions and get the following information before proceeding with the test:

1. **Local Card Up** - If you are not sure if the local card is up, run *osiadmin* and select "LAN (802.3 or FDDI)" from the menu. Next select "Start LAN (802.3 or FDDI)" from the menu. This will restart the card. Exit *osiadmin*.
2. **Remote Card Up** - Verify that the remote card is up using whatever utilities are provided by the remote node (for HP, use *osiadmin* as described above).
3. **Remote MAC Address** - determine the Media Access Control address of the remote node. For HP systems, this can be done on the remote through *osidiag* by selecting the "LAN 802.3 Tests" and then the Status menu item. The MAC address is labeled LAN Interface address, in the Status display. Exit *osidiag*.

Running the LAN Tests

1. Log on as `root` on the local node. (Omit if already logged on).
2. Perform the "Pre-Test Checklist," if not done already.
3. Start *osidiag* by typing `/etc/osidiag`.
4. Select "LAN Tests" from the main menu.
5. Create a result file which will contain a copy of the output produced by the LAN tests (should you encounter an error during the test execution an error during the test execution, you will have a record of the input and return values for the test operation. This may be useful in case you need to contact your HP support representative). The steps to create this are:
   a. Select *Utilities* from the LAN menu.
   b. Select "Open Result File" from the Utilities menu.
   c. Enter the name of the file you wish to hold the results (for example, /tmp/lan.res).

   d. Press the "Done" key (if you haven't already) to open the file.

   e. A message will be displayed indicating the file was opened. Press the space bar. If the message indicates some error occurred, check the name you specified.

   f. Press the "Previous Menu" key to return to the LAN Menu.

6. Select "Test Frames" from the LAN menu. This will generate special packets that the remote will recognize as requiring an acknowledgement.

7. Enter the device file name to issue the test from. The default will be sufficient unless you have multiple I/O cards.

8. Enter the value previously retrieved for the remote MAC address. This value must be entered in hexadecimal and is always 6 bytes (12 hex digits) long. Do NOT include the leading "0x" if present. Press the "Done" key.

   NOTE: the default value provided is your local MAC address, you must overwrite this with the remote's address.

9. Examine the field labeled test status in the output. If it says PASSED, then the two nodes can successfully communicate over the LAN. Proceed to the "Testing OTS Interoperability" section in this chapter.

   If *osidiag* says FAILED, then see the following subsection on "Interpreting LAN Errors."

10. At this point you will be in one of the following situations:

    - **LAN problem corrected** - If you have made a change that corrected your LAN problem then proceed to the OTS layer tests.
    - **Configuration change required** - If the corrective action in the following sections require you to change a configuration parameter, then do so and rerun the LAN test.
    - **Problem persists** - If you have been unsuccessful in correcting the problem, then gather the information you have collected to provide to your HP support representative. See the section "Submitting Problem Information to HP" in chapter 4.

Interpreting LAN Errors

open()

This attempts to open the LAN device. The "interface name" is extracted from the OTS configuration subnet configuration. If you fail to open the device you may have the wrong value configured (or no value configured). Check the parameter lan3_dev_name that is displayed at the top of the *osidiag* output and verify that this file exists and has appropriate permissions ("crw-rw-rw-"). On Series 800, type the command `lssf <device_file>`. If the file exists, you may not have LAN configured in your kernel correctly. See the System Administration manuals for information about configuring the LAN device. If you have recently installed LAN, then make sure you rebooted with the new kernel provided by update.

ioctl()

The *ioctl()* command is used to set up various options for your LAN access. When displayed in the operation field of the test status area, the type of *ioctl* command we were performing is displayed, (for example, "Changing to TEST frames (Super-user only)").

The following types of errors may be encountered:

- Changing to TEST frames. This operation is used to send TEST frames which are echoed back by the remote. By default, user information (data) frames are sent across the LAN. You must be super user to perform this. Login again as `root` if you get an error.
- Setting SSAP. This operation is used to set your local Link Service Access Point (LSAP). The "SSAP" means source SAP and "DSAP" means destination SAP. Don't confuse this with the SSAP referred to in Session layer tests. When running the expert level Send and Receive frame tests, set the SAP to "FE" (see the "General Concepts" section in chapter 9, "Addressing"). We will fail to set this SAP if it is currently in use by another *osidiag* process or by OTS. Check if OTS is running and if so, you must stop it.

read()

This operation is performed to receive information from the remote, either on the Receive or Test Frame operation. Typically this will indicate a timeout.

If you are performing the Test Frame operation, this indicates that the remote node you specified did not echo back the packet. This can occur for the following reasons:

- The MAC address you specified is incorrect. Recheck the address against that configured for the remote.

- There is no physical connection between the two systems. Check the cabling between the two systems. Also, if you have a router between the two systems, it may not forward TEST frames across networks depending on the configuration.
- The remote does not support IEEE 802.2 TEST frames. Some vendors do not support this operation. If the previous two problems are not at issue, check the vendor's documentation for their LAN product to ensure support for this operation. If this is the case then you should proceed to the OTS verification.

write()

This operation is performed to send information to the remote system. The same errors as described under *read()* apply to *write()* as well. The following additional error can occur:

- The local system is not configured to use IEEE packets. This can be checked and, if necessary, corrected (by the super user) using the *lanconfig(1M)* command.

Interpreting LAN Trace Files

LAN link tracing shows the layer 2 (Data Link) packets exchanged with the remote.

When tracing is invoked by *osidiag* it displays only OSI traffic by restricting the records to those with Source and Destination Service Access Points (SSAP's and DSAP's) of 0xFE. The first few lines of the trace show this filtering.

Because the protocol at this layer is connectionless, there are only two records you should see: Transmitted and Received data.

A typical LAN 802.3 record is shown below:

```
Transmitted 89 bytes via 802.3  Fri Aug 03 09:24:13:55267 PDT 1990
     pid=[ICS]  interface=[lan0]
     Dest: 08-00-09-01-96-87  Source: 08-00-09-01-96-87  Length: 00-4b
14:  fe fe 03 81 22 01 32 bc 00 48 00 00 07 68 70 69    ....".2..H...hpi
30:  6e 64 6f 6e 07 68 70 69 6e 64 6f 6e 00 97 00 00    ndon.hpindon....
46:  00 48 cd 01 00 25 e5 00 00 2c 01 42 c3 02 5d d5    .H...%...,.B..].
62:  c0 01 0b c4 01 01 c6 01 03 c1 05 64 69 61 67 74    ...........diagt
78:  c2 05 64 69 61 67 74 87 02 00 00               ..diagt.........
```

The record tells us that this was an outbound packet and that it was 89 bytes long. The time transmitted is given as well as the destination and source MAC address.

The data itself is displayed in hex as well as in ASCII. The first three bytes of an OSI packet should always be "FE FE 03." This gives the SSAP, DSAP and the Link layer PDU type (data).

The fourth byte indicates the Network layer PDU type. This will be either 0x81 (CLNP) or 0x82 (ES/IS). The former carries upper layer information, the latter is used only for maintaining routing information.

The fifth byte gives the length of the Network layer PDU header. Any data comes after this length. The length includes bytes four and five. In this example, the Transport PDU begins at the 38th byte displayed. It has the value 0x25.

The second byte of the Transport PDU gives the type code. See the discussion on "Interpreting X.25 Trace Files" later in this chapter for a brief synopsis of the Transport type codes.

Testing X.25 Interoperability

The steps below describe how to test the connectivity at the Network layer for nodes connected via X.25.

Pre-Test Checklist

You need to ensure the following conditions and get the following information before proceeding with the test:

1. **Local Card Up** - If you are not sure if the local card is up, run the command *x25stat.*

   If the output says something like "All VC's inactive," or prints information about active connections, then the card is up and ready. Otherwise you will need to restart the card. This can be done through *osidiag* or *x25init.*

2. **Remote Card Up** - Verify that the remote card is up using whatever utilities are provided by the remote node (for HP, use *x25stat* as described above).

3. **Remote Stack Up** - This test relies on having an entity on the remote willing to accept an X.25 connection. The Transport Layer of the remote stack should always be in this state.

   Verify that the remote stack is up using whatever utilities are provided by the remote (for HP, use *otsstat*).

4. **Remote X.121 Address** - Determine the X.121 address which the remote OSI stack uses to accept connections. For HP systems, this is a two step process:

   a. Determine X.121 address for the card. Type the command `x25stat -c` to display this.

   b. Determine any subaddress used by the OSI stack. Use *osiadmin* and select "OTS," followed by "View Configuration," followed by "OTS Subnetwork Parameters" to display the configured values. Use the "Prev" and "Next" keys (next to the arrow keys) to move through the list. Look for "snet_x25_subaddress" to find the subaddress for the CONS or X.25 CLNS entity.

   The combination of the X.121 address for the card and the subaddress configured for the stack make the full X.121 address you should specify.

5. **Determine CLNS or CONS usage** - The X.25 link of the remote may be used over CONS or CLNS. Determine which it is. For HP systems, this can be done by using *osiadmin* to display subnetwork parameters as described in step 5b. CONS networks are identified with the header "snet_cons_x25," CLNS networks are identified with the header "snet_clns_x25."

Running the X.25 Tests

1. Log on as `root`. (Omit if already logged on).
2. Perform the "Pre-Test Checklist" above if not done already.
3. Start *osidiag* by typing "/etc/`osidiag`."
4. Select "WAN X.25 Tests" from the main menu.
5. Create a result file that will contain a copy of the output produced by the X.25 tests. (Should you encounter an error during the test execution, you will have a record of the input and return values for the test operation. This may be useful in case you need to contact your HP support representative.) The steps to create a result file are
   - **a.** Select "Utilities" from the X.25 menu.
   - **b.** Select "Open Result File" from the Utilities menu.
   - **c.** Enter the name of the file you wish to hold the results (for example, /tmp/x25.res).
   - **d.** Press the "Done" key (if you haven't already) to open the file.
   - **e.** A message will be displayed indicating the file was opened. Press the space bar. If the message indicates some error occurred, check the name you specified.
   - **f.** Press the "Previous Menu" key to return to the X.25 Menu.
6. Select "Connect" from the X.25 menu.
7. Enter the value previously determined for the remote X.121 address. This value should be both the address for the remote card as well as any subaddress concatenated into a single decimal number (for example, card address = 1111100, subaddress = 33, full X.121 address = 111110033).
8. Leave the X.25 Interface Name unchanged. This should reflect the name previously configured for OTS.

9. The protocol ID should be changed to the value 03010100, if you are accessing the remote stack over CONS. For CLNS, the protocol ID should be 81. The default value shown is used for the HP loopback test and should be overwritten.

10. Press the "Done" key after modifying the address and protocol ID.

11. Examine the field labeled "test status" in the last section of output. If it says PASSED, then the two nodes can successfully communicate over X.25. Proceed to the "OTS" verification section.

    If *osidiag* says FAILED, then see the following subsection on "Interpreting X.25 Errors."

12. Enable tracing by doing the following steps:

    a. Select "Utilities" from the Transport menu.

    b. Select "Tracing and Logging" from the Utilities menu.

    c. Turn the tracing on for X.25 by placing the letter "Y" after it. Leave all other fields set to "N."

    d. Press the "Done" key (if you haven't already) to cause the change in tracing to take effect.

    e. Press the "Previous Menu" key to return to the Transport Menu.

13. Rerun the connection test by selecting "Connect" and pressing the "Done" key when prompted for the addresses. Note that *osidiag* will preserve the addresses you entered from the previous run.

14. At this point, you will be in one of the following situations:

    - **X.25 problem corrected** - If you have made a change that corrected your X.25 problem then proceed to the OTS layer tests.

    - **Configuration change required** - If the corrective action in the following sections require you to change a configuration parameter, then do so and rerun the X.25 test.

    - **Problem persists** - If you have been unsuccessful in correcting the problem, then you will want to gather the information you have collected to provide to your HP support representative. See the section "Submitting Problem Information to HP" in chapter 4.

Interpreting X.25 Errors

If the X.25 test fails, *osidiag* will indicate which operation encountered the error. The following are the common operations that might fail.

socket()

If X.25 is not installed on your system, this operation will fail with a message indicating that the protocol chosen is not available.

bind()

If the OTS stack is running, *osidiag* may not bind to the same X.121 address which the stack uses. If supported by the switch, you may try binding to a different subaddress, or the NULL subaddress (if OTS is not using the NULL subaddress). Alternatively, you may need to stop OTS to run this test.

connect()

If you specify an invalid programmatic access name you will get an error on this call. Use the default value or leave it blank. If you believe that you have specified a correct access name then verify the value by issuing the "Status..." operation through the X.25 tests. The actual name is displayed under the "General Parameters" heading.

*recv(*OOB*)*

Most other X.25 errors are related to the receipt of an unexpected CLEAR or RESET packet. These messages come on "Out of Band" data. That is, it is outside of the normal X.25 data stream. Information describing the error is provided in more detail in the diagnostic code and meaning fields as well as the clear code and meaning field.

Some common diagnostics which may be reported are the following:

- (0) — No additional information - this may indicate that the request was delivered to the remote, but it was rejected by the OSI stack. One possible reason is the protocol ID. Make sure that for CONS you use 03010100 and for CLNS you use 81.
- (67) — Call Setup Problem - Invalid Called Address. This indicates the address entered is unrecognized by the switch (if one is used) or is not known to the receiving remote (for back to back). Check the address of the remote and correct.
- (231) — Connection Rejection - NSAP Unreachable. This may indicate that the subaddress you entered or the protocol ID entered were incorrect. Verify and change if necessary the protocol ID (03010100 for CONS, 81 for CLNS) and the subaddress.

If a loopback test fails, verify that your switch is capable of and configured for loopback operation. Specifically, the full X.121 address (address and subaddress) that you specify should be configured.

Interpreting X.25 Trace Files

X.25 layer tracing information shows the layer 3 (Network) packets exchanged with the remote.

osidiag invokes tracing via the *nettl(1M)* command.

The following are the packets typically found in such traces:

- Call Request - This is a request to establish an X.25 connection. The existence of such a log record indicates only that the request was sent out.
- Call Indication - This is displayed when the remote is attempting to connect to us. You should not see this unless you are running a loopback test or other X.25 network activity is taking place simultaneously.
- Call Connected - This indicates that your connection request was accepted by the remote. If you see this log record, you are successfully communicating with the remote at least through the network layer.
- Clear Indication - This indicates remote is terminating your connection. This is normal at the conclusion of the connection test. However if you see this without seeing a Call Connected, it indicates the remote (or the X.25 network itself) is rejecting your attempted connection. The "Diagnostic" displayed should match the one reported through *osidiag*.
- Clear Request - This indicates the local node is terminating a connection, or refusing to establish one. The diagnostic code can provide further information in the case of errors (241 is normal disconnect). See the *Troubleshooting X.25* manual for more information.
- Data Packet - This indicates the inbound or outbound transfer of data over an existing connection. The X.25 Connect test does not send data, but the upper layer tests will send data. Also note that the upper layer tests may multiplex over the same X.25 connection, so you will not necessarily see an X.25 connection for each Transport connection.

  The data carried on these packets for OSI communications are Transport Layer PDUs. The PDU type can be determined by the second byte displayed.

Here is a brief list of types:

Ex - connect PDU (where x is initial number of credits for flow control).

Dx - connect confirm PDU (where x is number of credits).

80 - disconnect PDU.

C0 - disconnect confirm PDU.

F0 - data PDU.

10 - expedited data PDU.

6x - acknowledgement PDU.

70 - error PDU.

4

Troubleshooting

Information and procedures for identifying and troubleshooting problems that you may encounter using HP's OSI products

Troubleshooting

This chapter provides information and procedures for identifying and troubleshooting problems that you may encounter using HP's OSI products. It describes the use of the diagnostic, status, and tracing and logging tools. It also contains references to other locations in this and other manuals that may be helpful.

Note If the problem you encounter occurs only when communicating with another system, see chapter 3.

Basic Steps

1. Interpret the initial error. Find more information about the specific error you encountered by looking in the location described in the next section, "Interpreting Errors."

2. Determine the status of components. Make sure all the OSI product components are up and running. See the section on "Checking System Status" later in this chapter.

3. Verify operation. If all components report that they are up and the problem persists, verify the ability of the link (X.25, 802.3 or FDDI), the stack, and the specific service you are using to communicate. See the section on "Running Verification Tests."

4. Gather more information. If the information from the basic verification test was insufficient to diagnose the problem, then you can get additional information through HP's tracing and logging facilities. See the section on "Collecting Troubleshooting Data."

5. Validate configuration. If you have not already done so, run *osiconfchk* to validate the configuration. This tool is described in chapter 5, "Using OSI Tools," of this manual. Also go over the possible problems listed in the section "Common Configuration Mistakes."

6. Validate installation. If the system behavior is still not correct, then check that your software installation has not been corrupted. The tool *pdfck* performs this task. See the section "Running pdfck."

7. Submit information to HP. If you were not successful in diagnosing and correcting the problem, contact your HP support representative. The last section in this chapter, "Submitting Problem Information to HP" describes what information you should collect.

Interpreting Errors

This section describes where you should look to find more information about an error you encountered.

Before looking for additional information, read the complete error text displayed on your screen. It may suggest corrective actions. Try any corrective actions and return here if the error persists.

The following list gives the places where you may encounter errors and how to interpret them.

- *osidiag* - You can find a description of how to interpret errors at the end of each section in chapter 3, "Performing Remote Interoperability Procedures." For example, if you ran an MMS local verification test and encountered an error, you would look at the end of the MMS section in chapter 3. More detailed information about *osidiag* can be found in chapter 7 of this manual.
- *nettl* log files - Not all errors that appear in log files are further described in other manuals. Many errors contain detailed text describing the problem. General information about reading log (and trace) messages is contained in chapter 6 of this manual. Detailed descriptions of OTS log messages are contained in chapter 8 of this manual. Some common OTS logged errors are described at the end of this chapter.
- User-written OSI applications - Errors you receive in applications you write are passed back through return codes. The return codes are described in the programmer's guide for the layer producing the error. You may also want to use the API tracing facility for the interface, also described in the programmer's guide (not available for MMS). The *nettl* log file may also contain information, or you may want to increase the logging level and reproduce the problem. See chapter 6, "Logging and Tracing," for more information on enabling tracing.
- FTAM interactive commands - These errors are described in the *HP FTAM/9000 User's Guide* and the *HP FTAM/9000 Reference Manual.*
- X.400 commands - For errors encountered in X.400-specific commands (for example, *x4start* and *x4loop*), see the *Managing X.400 Administrator's Guide.*
- OTS and OSI administrative commands - These are errors produced from

commands like *otsstart* and *osiconfchk*. Chapter 8 of this manual contains descriptions for errors from commands with the osi- and ots- prefixes. More information may also be in the log files in /etc/net/osi/ots: OTS.startlog, OTS.translog and OTS.updatelog.

- System panics - Panics occur when an irrecoverable error is detected by the HP-UX operating system. At reboot after a panic, a copy of the state of the panicked system will be saved by the command *savecore* that is typically invoked by your /etc/rc script. You may need to create a special directory typically named /tmp/syscore in order for *savecore* to actually save the image to disk. Interpreting panics is done by your HP support representative or the factory.

Checking System Status

There are several tools that allow you to check if the links, stack, and services are up and running. Perform the following steps to verify that your local system is up.

1. Run *otsstat* to verify the status of OTS/9000 and its attachment to the underlying links.
   - If the status of OTS is "NOT RUNNING," then start the stack using *otsstart* or *osiadmin*.
   - If the status of OTS is "NOT RUNNING" after issuing *otsstart*, then you typically have a configuration problem. Run *osiconfchk*. See chapter 5, "Using OSI Tools" for more information on *osiconfchk*. You can also check the files /etc/net/osi/ots/OTS.startlog and /etc/net/osi/ots/OTS.translog for more information.
   - If the status of OTS is "RUNNING" but a link is down, then proceed to step 2.
2. If you have configured one or more LAN (802.3 or FDDI) cards, run *lanscan* to show the status of all LAN cards on the system.
   - If the Net-Interface State shown is "DOWN," the card is disabled. Restart the card by running *osiadmin* or *ifconfig*.

Note The Encapsulation Method shown should include "IEEE." If it does not, then you should reconfigure the card by modifying the lanconfig entry in /etc/netlinkrc. Then issue the command `lanconfig` *`lanx`* `ether ieee` where *lanx* is the LAN interface number reported by *lanscan*.

   - If OTS still indicates the link as "DOWN" while *lanscan* indicates that it is up, verify that the configured interface name shown by *otsstat* matches that shown by *lanscan*.
   - If the card remains down after starting it, you may have a hardware or cabling problem.
3. If you have configured one or more X.25 cards, then run *x25stat* against each card's device file name (for example, *x25stat* -d /dev/x25_0).
   - If the message says "not currently active," you should start the card by using *osiadmin* or *x25init*.
   - If the message says "Level 2 is DOWN," there is a communication problem between the card and the switch. Check the cabling and the configuration of the device your X.25 card is connected to.

4. If you are using the FTAM or MMS Services, run *osistat* to verify that the shared memory segment is present and to display the number of currently active FTAM and MMS applications. See chapter 5, "Using OSI Tools," for more information on *osistat*.

   - If you are using FTAM, then the count of "Active FTAM responder service providers" should be one or more. If not, then run *osiadmin* or *osistart* to start FTAM.
   - If you are running MMS, the count for "Active MMS initiator service providers" can be zero if no MMS activity is currently taking place.
   - If *osistat* indicates that it is "Unable to access the shared-memory segment" you should examine the file /etc/net/osi/ots/OTS.startlog for any related errors. You may need to regenerate your kernel to increase the available amount of shared memory or semaphores.

5. If you are running X.400, then type `/usr/lib/x400/x4stat` to display the X.400 processes that are enabled.

   - If the process status of any component displayed is "not running!", then issue the command *x4start*.
   - Use the *Managing HP X.400 Administrator's Guide* if problems persist.

Running Verification Tests

The verification strategy described here follows a bottom up approach. It first checks the ability of the links to communicate, then the OTS Transport Layer, and last the desired service.

All the verification tests use *osidiag*. Testing the X.400 services may alternatively be performed through *x4admin* or the X.400 tools described in *Managing HP X.400.*

osidiag may be invoked directly from the command line, or through *osiadmin* by selecting "Test Connectivity->" for the appropriate layer.

Note Hewlett-Packard strongly recommends that you run these verification tests even if the problem you are troubleshooting is seen through a user-written application. If the verification tests fail, then there is a general network problem that you should correct. If these tests pass, then the problem is related to your program's behavior. The "Collecting Troubleshooting Data" section will describe how to further analyze program-specific problems.

Link Verification

For X.25, run the "Loopback..." test under *osidiag*'s "X.25 Test Cases."

- If you have multiple X.25 cards, it is easier to run this test under *osiadmin*, so that *osidiag* will use the correct default values.
- If OTS is not configured to use subaddressing, you will not be able to run this test when OTS is running. This happens when you see an "Address already in use" error. Try running the "Connect..." test instead, being sure the Protocol ID field is blank and not "dx25."
- If you encounter errors, see the discussion of "Interpreting X.25 Errors" in the "Performing Remote Interoperability Procedures" chapter of this manual.

For LAN, run the "Test Frames..." test under *osidiag*'s "LAN Test Cases."

- If you have multiple LAN cards, it is easier to run this test under *osiadmin*, so that *osidiag* will use the correct default values.
- If you encounter errors, see the discussion of "Interpreting LAN Errors" in the "Performing Remote Interoperability Procedures" chapter of this manual.

OTS Transport Verification

To verify OTS Transport, do the following:

1. Go to "Transport Tests-" under *osidiag.*

2. Select the "Loopback..." test.

3. By default, *osidiag* will run over the first LAN subnet configured. If no LAN is configured, then it will run over the first WAN subnet configured. Overwrite the Network Address if you want to test a subnetwork other than the default.

4. If you encounter errors, see the discussion of "Interpreting Transport Errors" in chapter 3, "Performing Remote Interoperability Procedures."

You should run this test for each subnetwork you have configured. The subnetwork used is determined indirectly by the network address you specify in step 3.

The other network address values can be found on your Local Configuration Worksheets. Alternatively, you can find the other local addresses by doing the following.

1. Select the "Status..." test from the "Transport Test Cases" menu.

2. Look for fields labeled snet_local_net_address. These are the local network addresses you have configured for your subnetworks.

Service Verification

The verification steps for the X.400, MMS and FTAM products are given in more detail in the respective "Installing and Configuring" manuals. The general steps for service verification are outlined below.

1. From *osidiag*, select the service to test. Or from *osiadmin*, select "Test Connectivity..." in the menu for the service to test.

2. Select the "Loopback" test from the menu. For FTAM, use "Connect" instead of loopback.

3. Use the default configured values that *osidiag* presents. For more information about a parameter, press the "Help" key when in the field in question.

4. If you encounter errors, see the discussion of "Interpreting Errors" for the specific layer in the "Performing Remote Interoperability Procedures" chapter of this manual.

Collecting Troubleshooting Data

If the information already given was not sufficient to isolate and correct the problem, use the tracing and logging facilities to gather more information.

The method you will use to gather tracing and logging information will depend on the error that is produced. The possibilities are listed below.

- *osidiag* - If the problem is reported during a run of *osidiag*, then use the tracing and logging utilities provided by *osidiag*. This procedure is described in the following subsection, "Tracing and Logging Through osidiag."
- X.400 - If the problem occurs through an X.400 application, then follow the tracing procedures described in the *Managing HP X.400* manual.
- User Application - If the problem occurs through a user-written application, then there are two facilities that can give you further information. This procedure is described in the subsection "Tracing and Logging User Applications."

Tracing and Logging through osidiag

osidiag provides the ability to automatically capture *nettl* trace and log information and append it to the output for a test operation. *osidiag* also allows you to copy the results displayed to the screen to a result file. The following steps describe how to enable both facilities through *osidiag*.

1. Invoke *osidiag* either directly or through *osiadmin*.
2. Go to the Utilities menu.
3. Select "Open Result File" and enter a file to save the results of the test operation (for example, /tmp/ftam.res).

Note The file specified will be overwritten if it already exists.

4. Select "Tracing and Logging."
5. Enable logging for OTS, by placing a "Y" after its name in the pop-up window. If you are testing FTAM or MMS, enable logging for that service as well as ULA.

   Enable tracing for the layer being tested and the layer below. So, for FTAM you would enable FTAM and ULA. For Transport, you would enable Transport and Network. Use the "Help" facility for more information about a particular flag.

6. Press "Done" on the Trace and Log screen after setting the appropriate fields to "Y."

7. Use the default values presented for the log and trace levels by pressing "Done" on these screens.

8. Exit the Utilities menu.

9. Now rerun the test you are gathering information for. The results of the logging and tracing will be displayed on your screen as well as copied to the file you specified in step 3.

See "Common Logged Errors" at the end of this chapter to analyze the errors logged.

The trace information produced may be useful to your HP support representative. Interpreting traces requires a good understanding of the various OSI protocol internals. Some information on trace interpretation is given at the end of the "Session," "OTS," "X.25" and "LAN" sections of chapter 3.

Tracing and Logging User Applications

Two facilities exist to provide more data about the behavior and errors encountered by your application, API tracing and *nettl*.

API Tracing - This facility allows you to trace the calls your application makes to the Application Programmatic Interface. The mechanism to perform API tracing is described in the programmer's guide for the service you are using.

nettl - The *nettl* command allows you to enable logging and tracing in the OTS stack as well as for the FTAM and MMS services. The syntax of the *nettl* and *netfmt* commands is given in chapter 6 of this manual, "Logging and Tracing."

Hewlett-Packard recommends that you enable warning and error logging for stack components at and below the service you are using. The table shown below shows the entities on which you should enable logging for gathering information.

Tracing is only recommended if you have a good understanding of the protocol internals. When used, it is useful to trace at the layer your application uses and one level below.

For instance, to trace an FTAM application, you might enable tracing for the FTAM entities and the ULA entities. To trace an XTI application, you might enable Transport and Network tracing. The kind of tracing will usually be PDU In, PDU Out, Header In, and Header Out.

Service/Layer	Entities to Log
X.400*	X.400 Logging facilities + OTS entities + Link entity
FTAM	FTAM_VFS FTAM_USER FTAM_INIT FTAM_RESP + ULA entities + OTS entities + Link entity
MMS	MMS + ULA entities + OTS entities + Link entity
ULA**	SHM ACSE_US HPS CM EM ULA_UTILS
APRI	ACSE_PRES + OTS entities + Link entity
OTS	SESSION TRANSPORT NETWORK OTS
LAN 802.3	NS_LS_DRIVER
X.25	NS_LS_X25***
FDDI	FDDI

* The tracing and logging of X.400 specific components can be done through *x4admin*, and is described in the *Managing HP X.400* manual. Tracing and logging of OTS and the link is still performed through *nettl* for X.400.

** ULA provides facilities common to FTAM and MMS. When tracing or logging MMS and FTAM these entities should usually be included.

*** The entity NS_LS_X25 is used only for logging. If you wish to perform tracing at X.25, use entity X25L3 for layer 3 communication, or entity X25L2 for layer 2.

See "Common Logged Errors" at the end of this chapter to analyze the errors logged.

Running pdfck

pdfck is a program designed to compare the file descriptions in a Product Description File (PDF) to the actual files on your file system. It is intended as a tool to audit the file system and detect corruption or tampering. X.400 users will use the utility *x4perm* described in the *Managing HP X.400* manual.

To run *pdfck* perform the following steps:

1. From the system prompt, enter *pdfck* with the following syntax:

   ```
   pdfck /system/fileset_name/pdf
   ```

   where */system* is a directory name (enter it as shown), and *fileset_name* is the name of the fileset to be checked. The following fileset names can be used with *pdfck*:

   ```
   APLI-MAN      MAP30        OTS-KRN        OTS-SES-PRG

   APLI-PRG      MAP30-MAN    OTS-MAN        ULA-RUN

   FTAM          OSIF-MAN     OTS-RUN        XTI-MAN

   MMS           OSIM-MAN     OTS-SES-MAN    XTI-PRG
   ```

   If *pdfck* passes, the root prompt is returned. If *pdfck* finds a problem it displays a message as shown:

   ```
   pathname:diff_field[(details)][...]
   ```

   where *diff_field* is one of the field names specified in *pdf*. The fields are pathname, owner, group, mode, size, links, version, checksum, and linked to. Each field is separated by a colon (:). For more details on *pdfck*, refer to the online HP-UX manpages on *pdfck* and *pdf*.

2. If there is a file problem found with pdfck, it usually indicates an installation problem. Verify that the software was installed correctly.

Common Configuration Mistakes

The following list describes some common configuration errors that may result in failure during verification. The errors are grouped by the configuration file which contains them.

See the "OTS and Related Parameters" chapter in the *Installing and Administering OSI Transport Services* manual for detailed information about all the parameters.

ots_dests

Destination system configuration is a common error and easy to correct.

- If you mistype the dest_net_address or the dest_phys_address, you will fail in your attempt to connect.
- Destination entries must be made in order for loopback to work over both CONS and CLNS/X.25. Be sure to include any X.25 subaddress when specifying the physical address.
- Destination entries must be made in order for loopback to work over CLNS/LAN when CLNP Subset 0 is used. In other cases, destination entries for LAN should be avoided.
- If you specify the incorrect outgoing subnetwork for a destination, you will fail to connect.
- If you specify reverse charging for X.25, the remote must be configured to accept it. Otherwise, it will reject your connection.

ots_subnets

- If you mistype your local address, remote systems will be unsuccessful in connecting with you.
- Incorrect CLNP subset for LAN. This can prevent interoperability with other systems on the LAN. See the parameter snet_clnp_subset.
- If you have a LAN based network with a large number of nodes (over 200), you should increase the parameter snet_max_es_entries. Not doing so can result in failure to establish connections with other systems.
- Incorrect interface name. Specifying an interface name that is not configured will result in warnings, but OTS will still start. If *otsstat* indicates any links are "NOT RUNNING," verify these names.

ots_parms

- If you are using close to the limit of connections available (448) through OTS, the ses_reuse_tp_con flag may hold open connections you need to recycle.
- If you are using CONS/X.25, ensure that the Transport class OTS requests is acceptable to the remote. The tpcons_pref_mux_class parameter determines whether you use class 2 or 4. The parameter tpcons_tp0_only forces class 0.
- Some vendors do not accept the OSI Transport protocol ID carried on X.25 connection requests. This can be disabled with tpcons_null_pid.

local_app

- Setting the maximum invocations, or inbound associations too low can result in your failing to receive or create connections with remote MMS or FTAM applications.
- If you manually edit these files rather than using *osiadmin*, you should be sure that all FTAM application titles match the ftam_ddn_lookup_path parameter.

remote_app

- In order to perform MMS loopback, you should configure local MMS applications as both local and remote applications.
- Mistyping a remote application title or alias for FTAM and MMS will result in your connection failing.
- Mistyping a remote presentation address for FTAM and MMS will result in your connection failing.

mms_parms

- Setting the version incorrectly may prevent interoperability.
- Setting the max segment size below 512 may prevent interoperability with earlier HP versions of MMS.

Common Logged Errors

The following list describes some of the more common errors you can encounter which are logged by OTS. These errors are sorted by the subsystem name and the message ID as it appears in the log. More information for some of these errors can be found in the "Messages" chapter of this manual.

Subsystem OTS:

[9600] High Access Method ERROR - These errors indicate an error in the streams interface between user programs and the OTS stack in the kernel. One common problem is attempting a bind operation to an invalid network address or SAP. Verify the local address used on your test and try again.

[9800] x25 ERROR - These errors may indicate that the X.25 card has not been started or has gone down. Use *otsstat* and *x25stat* (or their equivalents under *osidiag*) to view the status of X.25. Restart X.25 if necessary under *osiadmin*.

Subsystem ACSE_PRES:

[6010] P_DCNX_KO - This error indicates that the presentation layer is releasing the connection after some error was encountered. Examine the log or user application for other errors which occurred.

[6011] PST_LOG - These errors indicate a problem found by the presentation layer in the OTS stack. This may indicate a protocol error, or an ASN.1 encoding issue. Examine the packets sent in the trace output.

Subsystem SESSION:

[5008] S_REJECT - This error indicates that a session connection was not successfully established. If the log does not show a T_REJECT error, this indicates that the problem was at the session layer. If a T_REJECT error is logged, then the S_REJECT is just a propagation of a Transport (or lower) layer problem.

[5010] S_DCNX_KO - This error is logged after the Session layer has successfully established a connection, but is releasing the connection in some error state. Examine the log or user application for other errors which occurred.

Subsystem TRANSPORT:

Note Transport errors will have message IDs that begin with either 41xx or 40xx, depending on whether you are running over CLNS or CONS respectively. The statements shown below apply as well to the 40xx errors.

[4103] **Protocol Error** - This error indicates a protocol error was detected at this layer. You should verify that both sides are using correct combinations of Class, Alternate Class, Flow Control and Expedited Data. If problem persists, you should ensure that you have both Transport and Network layer traces of the dialog, and contact your HP support representative.

[4108] **T_REJECT** - This error indicates that the Transport connection was not successfully established. You should examine the log for other errors (such as routing problems, or network layer problems). If no other logged errors appear, the problem may be that no process is listening at the address you specified. Verify the remote address given as well as what addresses are available at the remote.

[4113] **Routing** - This error indicates that CLNS was unable to determine the NSAP/MAC address translation for the destination Network Address given. Possible problems are: you mistyped the address, the remote is not up, the remote does not support ES/IS, the remote's destination configuration (ots_dests) is incorrect.

[4115] **Processing Error** - This error usually appears in conjunction with a protocol error. It may give more information about what was not liked with the PDU.

Subsystem NETWORK:

[3003] **N_REJECT** - This indicates the X.25 connection could not be established with the remote. Possible problems are that the NSAP/X.121 address mapping for this remote is incorrect. Facility negotiation failed with X.25 (for example, Reverse Charging, X.25 '80/'84). The remote stack or card may not be enabled. The remote system may not be able to accept anymore X.25 connections.

[3014] **N_ERROR** - This may provide more detailed information for why the network connection was rejected. Errors regarding route resolution mean that your destination address configuration is in error.

Submitting Problem Information to HP

The following items will be very useful in having HP Support diagnose problems with your OSI products.

- **Result Files** - The output created for all your runs of *osidiag*. Examples of the recommended file names are /tmp/ftam.res and /tmp/tran.res.
- **OTS Configuration Files** - The contents of the following files in /etc/net/osi/conf:
  - **local_app** - For MMS for FTAM problems only.
  - **mms_parms** - For MMS problems only.
  - **ots_dests** - For nodes running over X.25 or NULL internet.
  - **ots_parms** - In all cases.
  - **ots_routes** - In all cases.
  - **ots_subnets** - In all cases.
  - **remote_app** - For MMS or FTAM only.
  - **ap_user_app** - For APRI problems only.
  - **cm_user_app** - For CMIS problems only.
- **X.400 Configuration Report**. If you are using X.400 and experience problems, type the following: `/usr/lib/x400/x4dump`. A file called /tmp/x4dump.sh is created which can be e-mailed or put on tape.
- X.25 Configuration Report. If you are using X.25 and experience problems, type the following: `/usr/bin/x25stat -c /tmp/x25.cnf`. Print the resulting file.
- **LAN Configuration Report**. If you are using LAN (802.3 or FDDI) and experience problems, type the following: `/etc/lanscan /tmp/lan.cnf`. Print the resulting file.

5

Using OSI Tools

The software tools you'll use to configure, maintain, and troubleshoot your HP OSI products

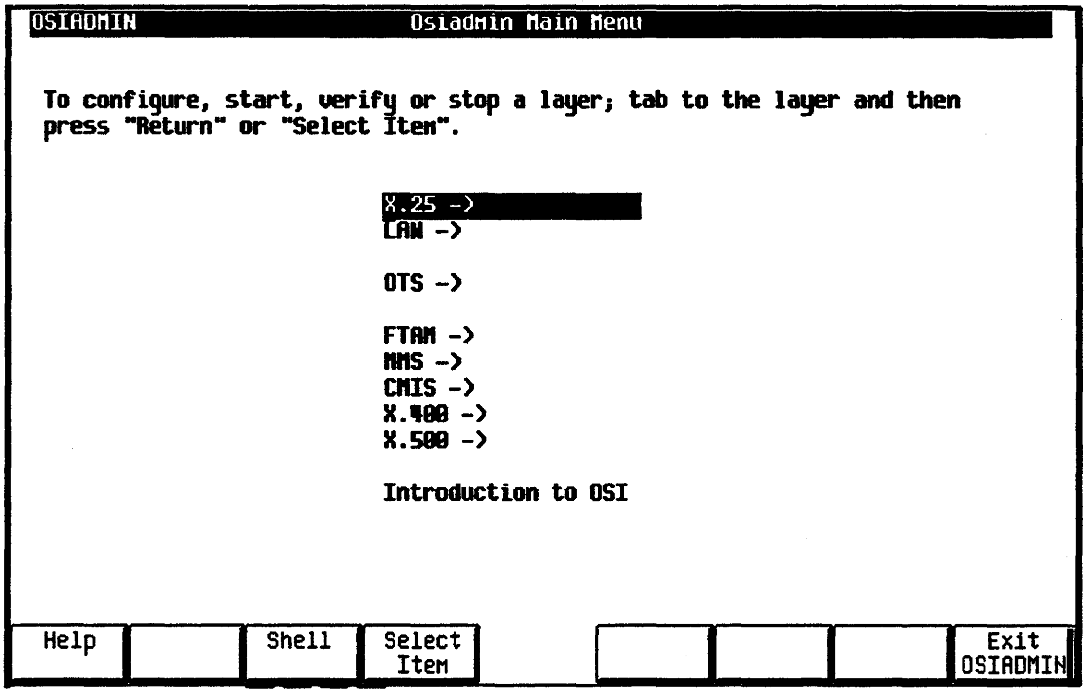
Using OSI Tools

This chapter briefly describes the software programs and commands you'll use to configure, maintain, and troubleshoot your HP OSI products.

OSIADMIN

The *osiadmin* (OSI Administration) program gives you access to the various configuration, administration, and diagnostic tools you need to setup and maintain your HP OSI products. *osiadmin* acts as an umbrella for these tools, providing you with a single, easy way to configure, start, stop, and test each product.

The screen below shows the HP OSI products you can administer through *osiadmin*.



```
OSIADMIN                    Osiadmin Main Menu

To configure, start, verify or stop a layer; tab to the layer and then
press "Return" or "Select Item".

                         X.25 ->
                         LAN ->

                         OTS ->

                         FTAM ->
                         MMS ->
                         CMIS ->
                         X.400 ->
                         X.500 ->

                         Introduction to OSI
```

The following table shows the HP OSI products, administration functions, and software tools called by *osiadmin*.

Product	**Function**	**Tool**
LAN	configure	SAM (System Administration Manager)
	start	ifconfig
	stop	ifconfig
	test	osidiag
X.25	configure	SAM
	start	x25init
	stop	x25stop
	view	x25stat
	test	osidiag
OTS	configure	osiconf
	start	otsstart
	view	osiadmin
	test	osidiag
	update	otsupdate
FTAM	configure	osiconf
	start	ftam_resp
	stop	ftam_resp
	view	osiadmin
	test	osidiag
MMS	configure	osiconf
	view	osiadmin
	test	osidiag
X.400	configure	x4admin
	start	x4start
	stop	x4stop
	view	x4configprint
	test	osidiag
	administer	x4admin

Using osiconf to Manage Your Configuration

The *osiconf* program is an interactive configuration tool you can use to do the following:

- Verify that your configuration information has been entered with the proper syntax and range, and that it is consistent between OTS FTAM, and MMS.
- Modify the OTS FTAM and MMS configuration files.

osiconf is most commonly used through *osiadmin*, but may be started alone. *osiconf* has two "modes." In update mode, you can use *osiconf* to configure specific OTS parameters in either one of modes: dynamic or restart (the default setting). If *osiconf* is set to dynamic mode, only dynamic parameters can be modified. When your modifications are complete, changes take effect when you exit *osiconf*, or later by using the *osiupdate* program. When dynamic changes take effect, they are applied to the running OTS, FTAM, or MMS.

In restart mode, you can modify all parameters in *osiconf*. Changes take effect when you use the *osistart* program (if this is the initial configuration), or when you reboot (if you've changed existing configuration).

For more information, refer to the online HP-UX manpage for *osiconf*.

Running osiconfchk

The *osiconfchk* program is a configuration tool that allows you to verify the correctness of configuration files prior to actually starting the stack. Checks are performed to ensure that all required information is present, that parameters are of the proper syntax and range, and that information is consistent across configuration files.

Note For X.400, *osiconfchk* checks only those parameters that are also configured in OTS. The *osiconfchk* program does not verify the correctness of the X.400 configuration.

The directory where the configuration files may be found is specified by setting the environment variable OSI_CONFIG. If OSI_CONFIG is not set, /etc/net/osi/conf is searched as the default location.

To run the *osiconfchk* program, perform the following steps:

1. Login as `root`, then type `osiconfchk`.

   For details on the options and variables, refer to the HP-UX online manpages for *osiconfchk*.

2. If there are no parameter problems, the root prompt re-appears.

3. If *osiconfchk* finds problems with the configuration parameters, it displays the file the problem was in, the line of the file, and the actual parameter at fault. Examples 1 and 2 show two sample outputs of *osiconfchk* errors.

   Example 1

   ```
   ::::::::::::::
   /etc/net/osi/conf/ots_parms
   ::::::::::::::

   Line 43:
   ->tpcons_max_tpdu_size        2047    #       128        8192            2048
   Integer value is not a power of 2 ( 128, 256, 512, etc. )  (CHK021)
   ```

   Example 2

   ```
   ::::::::::::::
   /etc/net/osi/conf/local_app
   ::::::::::::::

   Line 26:
   ->ae_local_paddr    7874693.0001.0001      # p-, s-, and t_selector
   An even number of hexadecimal digits is required.  (CHK023)
   ```

   The filename is a configuration file from X25, OTS, FTAM, MMS, X400, or CMIS. Line number is the line in the file. The second line is the actual parameter name and value of the parameter. The last line is a statement of what *osiconfchk* found wrong with the parameter.

4. Compare this display with the parameter value on the appropriate worksheet to be sure it was entered correctly.

   If everything matches, contact the network administrator for instructions or assistance.

OSIDIAG

The *osidiag* (OSI Diagnostic) program allows you to verify that the OSI services provided by the HP network node are operational. You'll use *osidiag* throughout the remote interoperability procedures. It can also be used as a layer troubleshooting aid in determining the functionality of the suspect component. You can start *osidiag* from within *osiadmin* or directly from the HP-UX system prompt. For more details on *osidiag* and the various tests it can perform, refer to chapter 6 of this manual.

osidiag provides tests for the following components of the OSI stack:

- FTAM Initiator Testing - This portion of *osidiag* tests the communication between HP's FTAM Initiator and the specified remote responder.
- MMS - This portion of *osidiag* allows you to verify the ability to create and accept MMS connections. Several other MMS utilities are also available.
- X.400 - Four tests are available for X.400. The RTS connection tests provide verification of HP's ability to establish or receive connections at the RTS layer. The User Agent connection test verifies HP's ability to send an X.400 message and receive a delivery confirmation from the remote node. In order to provide more feedback to the user, this test will track message activity as the message transfers from one X.400 process to another. The RTS loopback test does a local loopback test to verify that the RTS can communicate with OTS.
- ACSE/Presentation - This portion of *osidiag* allows you to create or receive ACSE/Presentation connections and send data.
- ROSE - The ROSE tests allow you to create or receive connections over ACSE/Presentation and issue and receive operation invocations.
- CMIS - This portion of *osidiag* allows you to create and receive CMIS connection. Basic network management functions may also be sent.
- Session - This portion *osidiag* provides verification of HP's ability to establish or receive connections at the session layer. Data, expedited data, and some of the activity management primitives may also be sent on this connection.
- Transport - The Transport portion of *osidiag* provides verification of HP's ability to establish or receive connections at the transport layer. Data, and expedited data may also be sent on this connection. Please read following note.

Note An entity must be running on the remote node, capable of receiving (or initiating) a session or transport level connection. For the HP node, *osidiag* provides the capability to both establish and receive the connection. However, some vendors may not have exposed access to this layer or the functionality provided by *osidiag*. In that case, the test desired may have to be run manually (without *osidiag*).

- X.25 - *osidiag* allows an X.25 connection to be established and brought down verifying layer 3 connectivity with the remote node. Card status and statistical information can also be reported.
- LAN (802.3 and FDDI) - This portion of *osidiag* uses the 802.3 Test Frames to determine if there is layer 2 connectivity. Card status may also be reported.

With *osidiag*, you can automatically start tracing and logging while running diagnostic tests at a specified layer of the stack. By designating a "Result File," you can store a duplicate of the information displayed while the test case is running. This file can be useful in relaying problem information if consulting help is requested.

Starting and Stopping OSI

Occasionally, while configuring, verifying, or troubleshooting any of the OSI products, you'll need to start or stop the OSI products. Use the commands described here.

osistart Command

The *osistart* command is executed from the system prompt. It can be used to start services, such as FTAM and X.400 along with the OTS component and whatever links are being used. This is accomplished by issuing the *osistart* command with no options.

The services can be started separately when osistart is used with the -s option. This also starts the OTS and link components (if necessary). For example:

```
osistart -s x400
```

For more details on the *osistart* command and its options, refer to the HP-UX online manpages.

Note After initial configuration, OTS/9000 is automatically started when the system is booted. Thus, OTS is usually up and will rarely (if ever) be started by *osistart*.

osistop Command

The *osistop* command is executed from the system prompt. It can be used to stop any services, such as FTAM and X.400. Using *osistop* with no options stops all OSI services.

The *osistop* command with either "ftam" or "x400" in conjunction with the `-s` option stops just the service designated, for example:

```
osistop -s ftam
```

For more details on the `osistop` command and its options, refer to the HP-UX online manpages.

osistat Command

Super-users can use the *osistat* command to get the status of various aspects of OSI products, for example, active service providers, aborts sent, aborts received, and connections established. Executed from the system prompt, the *osistat* command with no option prints global service provider process (SPP) statistics for FTAM and MMS. With the `-m` option, *osistat* prints memory buffer statistics without SPP statistics.

6

Logging and Tracing

Using *nettl* and *netfmt* to provide diagnostic information at various levels within the OSI stack

Logging and Tracing

In chapter 3, "Performing Remote Interoperability Procedures," the *osidiag* utility was used to perform various tests and help isolate problems. During the tests, *osidiag* made use of the logging and tracing facilities to provide diagnostic information and status at various levels within the stack. Logging and tracing was started by *osidiag*, and then terminated when that session of *osidiag* concluded. There may be times when you wish to use the logging and tracing facilities outside of *osidiag*. This section discusses the use of such utilities for the OSI products.

Hewlett-Packard OSI products use *nettl* to create log and trace files. The following discussions and procedures describe the method for generating and viewing trace and log data from the OSI products.

Note Troubleshooting connectivity problems can be a very complex area of network administration, requiring expert knowledge of both OSI networking and implementation. Refer to chapter 4, "Troubleshooting," for more information.

Also, accessing and interpreting log and trace files at the local node can aid in resolving problems on the local node. If the problem is determined to be at the remote node, the log and trace files at that node should also be examined. If the remote node is a Hewlett-Packard system, the tracing and logging guidelines presented in this chapter apply. If the remote node is another vendor, refer to that vendor's administrator or documents for help.

Logging and Tracing the OSI Stack

HP OSI logging and tracing is accomplished primarily through two utilities: *nettl* and *netfmt*. *nettl* can be used to control trace and log data collection for each subsystem configured on the system. The collected data is then organized and formatted with the *netfmt* command.

See the "nettl(1M)" and "netfmt(1M)" sections later in this chapter for more information regarding these utilities. (An HP-UX online manpage also exists for each of these commands.)

The amount of tracing and logging performed by the system can be controlled on a per subsystem basis. A subsystem is defined as a collection of software routines that share common functionality or facilities. The groups of subsystems listed below are contained within the OSI stack. Each subsystem is capable of producing trace or log data when enabled by the *nettl* command. To see what subsystems are configured on a system, type `nettl -status`.

Link Subsystems

The following subsystems make up the links that can be used by the OTS Stack.

- NS_LS_DRIVER - This is the IEEE 802.3 LAN link.
- FDDI - This is the FDDI LAN link.
- NS_LS_X25 - This is the X.25 link logging subsystem. Unlike other subsystems X.25/9000 has a subsystem dedicated to logging.
- X25L2 - This is the level 2 X.25 link tracing subsystem.
- X25L3 - This is the level 3 X.25 link tracing subsystem.

The error codes, tracing and logging procedures, and descriptions are listed in the link product manuals, *Installing and Administering LAN/9000* (P/N: Series 300: 98194-90015; Series 800: 98194-90016), *Installing and Administering FDDI/9000 Software* (P/N: J2156-61001), and *Installing and Administering X.25/9000* (P/N 36940-90004).

OTS Subsystems

The following subsystems are within the OTS Stack.

- NETWORK - This is the Network layer entity of the OTS stack. This subsystem includes both the Connection Oriented (CONS) and Connectionless (CLNS) Network Service entities.
- TRANSPORT - This is the Transport layer entity. Some CLNS logged errors may also appear under this entity.
- SESSION - This is the Session layer entity.
- ACSE_PRES - This is the ACSE/Presentation entity.
- OTS - This corresponds to non-layer specific tasks performed while managing the various stack entities, for example, communication between the OTS stack and user space, communication between the OTS stack and network devices, and other administrative tasks such as buffer management.

ULA Subsystems

The following subsystems make up the Upper Layer Architecture (ULA).

ULA provides ACSE/Presentation services to FTAM and MMS, as well as other support facilities.

- SHM - This is the shared memory access routines. These are used by both the User Process (UP) and the Service Provider Process (SPP). Problems here are typically propagations of earlier errors (check the log instance). If the initial error is with the SHM then the stack may not have been brought up correctly (using *otsstat*).
- HPS - This is the Host Presentation Services component of the Service Provider Process (SPP). These routines deal with the Presentation layer as well as access to the Session layer of OTS/9000. Errors here typically indicate some OTS type problem (for example, addressing, unexpected aborts, or some other lower layer connectivity problem).
- CM - This module of the Service Provider Process (SPP) deals with Connection Management.
- ULA_UTILS - These are various utility procedures within the Service Provider Process (SPP). Errors logged here may indicate that the SPP has encountered an exceptional condition (for example, signal).

- EM - This is the Event Management component of the User Process (UP) and Service Provider Process (SPP). This portion of the code deals with control of messages passed between the UP and SPP. Problems logged by EM are typically propagations of earlier errors (check the log instance).

MMS Subsystems

The MMS subsystem includes both the MMS library and the MMS service provider process.

FTAM Subsystems

The following subsystems are within the FTAM stack.

- FTAM_INIT - The FTAM Initiator is the FTAM code that resides in the Service Provider Process (SPP). This code deals with the encoding and decoding of FTAM PDU's. It also coordinates the behavior of the other modules in the SPP.
- FTAM_RESP - The FTAM Responder is a daemon process, "/etc/ftam_resp." This process is made up of the same components as a SPP, however it does not communicate with a User Process. Log messages indicating this entity are generated by the code that deals with encoding and decoding responses to requests from FTAM Initiators.
- FTAM_VFS - The FTAM Virtual File Store is a component of the FTAM Responder process. The file store deals with accessing the files on the system, as well as enforcing access control to the file store.
- FTAM_USER - This is the FTAM library code that is linked in with the user library. It deals with passing messages between the User Process (UP) and the Service Provider Process (SPP). Errors reported by this layer typically indicate errors propagated up from the SPP. Another possibility is detecting the death of the SPP process (for example, from the *kill(1)* command or a fatal error). Finally, if there is an installation problem, the library may not be able to successfully communicate with the SPP.

X.400

The X.400 product does not use *nettl* or *netfmt* for tracing and logging. Instead, it uses two other utilities, *x4eventlog* and *x4logview*, for generating and formatting trace and log data.

X.400 critical errors are always logged to special X.400 log files. To view the X.400 critical error log files use *x4logview*.

"Event logging," or tracing of message activity, is initially enabled but may be disabled using the *x4eventlog* command. To view the eventlog files you can use the *x4logcat* utility or the HP-UX *more* command since the eventlog files are ASCII text. It is recommended that event logging be enabled when troubleshooting problems.

Refer to the *Managing HP X.400 Administrator's Guide* for more details on X.400 logging and tracing.

Network Logging

The *nettl* logging facility records unusual or exceptional events such as errors, warnings, and state transitions. Logging is part of standard network operation and should be started each time the system is booted.

Starting Logging

The following command should be placed into the /etc/netlinkrc file to enable logging for all subsystems:

```
nettl -start
```

You can verify that tracing and logging has been started by issuing the following command:

```
nettl -status
```

Series 800 only: For X.25/9000, the *nettl -start* command must be followed by the *nettl -fm 0 -c /dev/x25_0* command to initialize the firmware for tracing. This must be done manually when you install the software.

Altering Log Class

To temporarily alter the log classes enabled for a subsystem or group of subsystems the *nettl -log* command may be used. When logging for a subsystem is changed from the default class the new log classes will remain enabled until another *nettl -log* command is issued or until *nettl* is restarted (as in the case of a reboot). For example, the following command will enable all classes of logging for the SESSION subsystem:

```
nettl -log informative warning error disaster -entity SESSION
```

You may also use the abbreviated version:

```
nettl -l i e w d -e session
```

The type of information which the log file will contain about a given subsystem is determined by its log class. The meaning of the four log classes follows.

Disaster Log Class Messages

Disaster log class messages indicate an event that may jeopardize the system or network integrity. When a disaster log class message occurs the node should be taken offline and all networking operations aborted or suspended until the problem is corrected.

Error Log Class Messages

Error log class messages indicate an event that will not affect the overall system or network operation, but will cause an application program call to fail or complete with an error. An error event requires special action on the part of the user or application, for example, repeating a transmission request or reestablishing an SVC.

Warning Log Class Messages

Warning log class messages indicate an event that is recoverable by the subsystem. This may be an incorrectly specified parameter (and the default value is used), or a misuse of a command. Subsystems can recover from a warning event with no further action on the part of the user or application.

Information Log Class Messages

Informational log class messages describe significant events that cause a state change within a subsystem. These events do not require any exceptional action on the part of the subsystem, and are part of normal operation. These events include the establishment and termination of connections.

You can permanently modify the default logging options for a subsystem by using the *nettlconf* command. Permanently changing the log levels from their factory defaults is not recommended. A discussion of *nettlconf* and the *nettl* subsystem configuration file, /etc/conf/nettlgen.conf, can be found at the end of this chapter. There is an HP-UX online manpage for *nettlconf* and *nettlgen.conf*. As an example, the following command will permanently enable all classes of logging for the SESSION subsystem:

```
nettlconf -c 12 -o ss_OTS_go -i 93 -n SESSION -g "HP OSI MGMT" -l
libotsfmt.a
```

Any changes to the *nettl* configuration file require that the tracing and logging facility be reinitialized with the `nettlgen` utility. There is an HP-UX online manpage for this command.

Log Files and Logging Operations

When the *nettl* daemons are installed, the /etc/conf/nettlgen.conf file is used to configure the tracing and logging facility. This file contains network configuration information used by the *nettl* daemons to log and trace the activities on the network. Among other things, the file identifies the subsystems that are logged and what kind of information is logged. Entries are added to this file automatically when you install your software. However, you must run the *nettlgen* utility to reinitialize the tracing and logging facility after any changes have been made to the subsystem configuration file, /etc/conf/nettlgen.conf.

The default setting of the logging facility of *nettl* produces disaster messages from any installed subsystems.

The *nettl -stop* command stops the *nettl* daemons and terminates tracing and logging for all network subsystems. It is not recommended that you stop *nettl* because you will then lose any records of network disaster messages which might occur during that time.

Upon startup the *nettl* daemons open the log files specified in the /etc/conf/nettlgen.conf file. By default the log files are /usr/adm/nettl.LOG00 and /usr/adm/nettl.LOG01. This can be changed by modifying the /etc/conf/nettlgen.conf file, and running the *nettlgen* utility to reinitialize the tracing and logging facility. Even if you change the name of the log files, they will always have the `.LOG00` and `.LOG01` suffixes. The *nettlgen* command is described in the *HP-UX Manual Reference Pages*.

The *nettl* daemons always write to the `.LOG00` file. When that file is full, the daemons rename it to the `.LOG01` file and the *nettl* daemons create a new `.LOG00` file. If the `.LOG01` file already exists, the contents are destroyed. The *nettl* daemons continue writing to the `.LOG00` file.

This process continues writing to the `.LOG00` file and copying to the `.LOG01` file as long as the *nettl* daemons are running. This insures that the oldest active log data on the system is always in the `.LOG01` file, and the latest log data is always in the `.LOG00` file. The entries in the log files are correctly sequenced even across system shutdowns and reboots. The maximum size for these files is 500Kbytes by default, but this too can be changed by editing the /etc/conf/nettlgen.conf file, and running the *nettlgen* utility.

Each log entry is written in an internal binary format containing header information and log event data. The binary log data usually contains subsystem specific information describing the log event. If the log event originates from any of the OSI subsystems, the error code contained in the message can be referenced in chapter 8, "Messages."

Formatting and Viewing Log Messages

The binary log data contained in the log file must be converted to ASCII text with the *netfmt* utility. This is referred to as "formatting". *netfmt* formats the data in a human readable form suitable for viewing at a terminal screen or printing.

To format the default log file to the screen, enter the following:

```
netfmt -f /usr/adm/nettl.LOG00
```

netfmt produces a header banner and the log event data for each log event recorded in the log file. The header banner reflects the information contained in the binary log event header. A typical log header may look like:

```
*********************************HP OSI MGMT****************************@#%
  Timestamp           : Tue Dec 31 1991 10:05:23.815036
  Process ID          : ICS:                Subsystem        : 94
  User ID ( UID )     : -1                  Log Class        : DISASTER
  Device ID           : -1                  Path ID          : -1
  Connection ID       : -1                  Log Instance     : 0
  Location            : 00123
```

The following is a summary of the header fields.

Timestamp	The time when the log message was recorded.
Process ID	The ID of the process that recorded the log event.
Subsystem	The subsystem that recorded the log event.
User ID	The ID of the user that recorded the log event.
Device ID	The logical unit or card id of the network card.
Path ID	An application specific connection identifier.
Connection ID	A connection identifier.
Log Instance	A unique identifying number assigned to a specific log event. If several log events have the same log instance then the first logged event with that log instance is most likely the source of the event.
Location	Unique position within the subsystem that produced the log event. This is subsystem dependent.

Log Class	The severity of the log message. Log class can be one of 4 values: disaster, error, warning, or informative.

netfmt allows the user to filter out unwanted log messages. There are two types filters available in *netfmt*: global filters, which are effective on all subsystems, and subsystem filters which are effective on only specific subsystems. Filtering will be discussed later in this chapter.

The *netfmt* command can format only the information that has been logged by the *nettl* daemons. To change the information being logged, issue the *nettl* command with the `-log` option set, as shown in the X.25 example below:

```
nettl -log disaster error warning -entity ns_ls_x25
```

The command shown above causes warning messages in addition to the error and disaster messages to be logged. The syntax and semantics of the *nettl* and *netfmt* commands are described later in this chapter.

Network Tracing

Tracing is a detailed examination of operations performed by a subsystem. Where logging records unusual or exceptional events, tracing records normal operational events including the reception and transmission of data. Unlike logging which is part of standard network operation, tracing is used only as a debugging and troubleshooting tool and is not part of standard operation of a subsystem.

Starting Tracing

To start a trace, the *nettl* daemons and network logging must be active. Use *nettl -status* to verify these are running. The following is an example command to enable tracing for Session PDUs and PDU header information:

```
    nettl tn hdrin hdrout pduin pduout -e SESSION -f /tmp/sess
```

When tracing begins, two additional *nettl* daemons begin execution. If you subsequently issue a *ps* command, you will see four processes: two shown as `ntl_reader` and two shown as `nktl_daemon`. One pair of *nettl* daemons is dedicated to network logging and one pair is concerned with tracing. There may also be a *netfmt* process running to send disaster log messages to the system console.

Note Only one trace of an X.25/9000 card can be active at any given time.

Trace Files and Tracing Operations

The *nettl -traceon* command allows you to specify the files used in the trace, their size, and the maximum length of trace records. When tracing the `nettl` daemons use the same circular file method as used when logging. The pathname you specify in the command is used but a suffix is added so that the filenames have the following format: ***filename*.TRC*x*** (where *x* may be 0 or 1).

The *nettl* daemons write to the *filename.TRC0* file. When that file is full, the daemons rename the file to *filename.TRC1* and create a new *filename.TRC0* file. If the *filename.TRC1* file already exists, the contents are destroyed.

The process of writing to the *filename.TRC0* and renaming it to the *filename.TRC1* continues for the duration of the trace. This process insures that the oldest trace data on the system is always in the `.TRC1` file, and the latest trace data is always in the `.TRC0` file.

If you do not specify a trace file name, trace records are written to the standard output file, usually the terminal, in binary format. Use *netfmt* to convert the file to a readable format.

The *nettl* daemons capture trace records in a trace buffer when they are received from a network subsystem. The daemons store the records there until they can write them to the trace file. In some cases when large trace records are being produced very quickly or even when the system and the disks are under heavy load it is possible to lose trace records. To prevent this from occurring you can increase the size of the trace buffer with the `-size` option.

Each trace entry is written in an internal binary format. It contains a header information field and a data field. The header information includes a time stamp, a subsystem ID, a trace kind ID, and other miscellaneous fields. The data field contains the actual data that was transmitted or received.

Formatting and Viewing Trace Data

Use the *netfmt* command to view the trace data. This command formats the data in a human readable fashion that is suitable for viewing at a terminal screen or printing. The trace files can contain messages for all network subsystems running on the machine. The *netfmt* command allows you to filter out messages in which you are not interested.

Because tracing is primarily used in a troubleshooting or debugging situation, people using tracing typically want to see the trace data as it is created and act on it immediately. For this reason trace data is often piped immediately to the *netfmt* command. For example, consider the following command for LAN:

```
nettl -traceon pduin pduout -entity ns_ls_driver | netfmt -c
lantrace_filters
```

The commands shown above causes the *nettl* daemons to collect all incoming and outgoing packet data from the LAN card, and to write the data to the standard output file. The *netfmt* command reads the trace data from the standard input and writes the filtered and formatted records to the terminal. The filters are specified in the `lantrace_filters` file.

Because of the special relationship between the *nettl* daemons and *netfmt*, the shell is active when the *nettl* command is piped to *netfmt*. This is caused by the fact that the trace data is not produced by the *nettl* command, but rather the *nettl* daemons. The *nettl* command starts the *nettl* daemons and exits, when the *nettl* command exits the shell begins operation.

To turn off tracing use the *nettl -traceoff* command.

nettl(1M) Syntax

nettl controls the network tracing and logging facility. (This command requires super-user capability.)

Syntax

```
[-st]
[-sp]
[-ss level]
[-l class -e subsystem [subsystem...] [-m pack_size]
[-tn kind [kind...]  -c  dev_name
     -e subsystem [subsystem]...[-tm maxsize]
     [-f tr_name ] [-s limit]]
[-tf -e subsystem [subsystem]...]
[-fm 0|1|2  -c dev_name ]
```

Options

-start	Starts the *nettl* daemon, initializes the tracing and logging facility, and enables logging for all subsystems. The *nettl* daemons run in the background and maintains the network tracing and logging system. Log messages are sent to the log file. The default name is `/usr/adm/nettl.LOG`*xx*. The suffix *xx* is 00 or 01 that will be appended onto the filename. The default logging class for all subsystems is `disaster` or 8. See the `-log` option. This option may be abbreviated as `-st`.

Note Hewlett-Packard strongly recommends that the tracing and logging facility be started before any other networking. Otherwise, log data may be lost. The *nettl -st* command should be placed in `/etc/netlinkrc` before any other networking commands. This is done automatically if you have configured your system with the System Administration Manager (SAM).

-stop

Stops the *nettl* daemon, terminates the tracing and logging facility, and disables logging for all subsystems. The network should not be operated without the *nettl* daemon running.

This option may be abbreviated as `-sp`.

-status *info*

Reports tracing and logging status for all subsystems known to the *nettl* daemons. The *nettl* daemons must be running when you issue this command. *info* specifies the type of information that is to be displayed. The supported values are:

`log`	for logging status information
`trace`	for tracing status information
`ALL`	for tracing and logging status information

If *nettl* has not been started, this command will produce an error indicating that nettl is not enabled. Otherwise you can assume that *nettl* is active.

This option may be abbreviated as `-ss`.

-traceon *kind*

Starts tracing on the specified subsystem. The *nettl* daemon must be running when you issue this command. *kind* defines the trace masks used by the tracing facility before recording a message. You may enter either the keyword or mask as the *kind* value. The supported values are:

KEYWORD	MASK	Meaning
`hdrin`	0x80000000	Header Received
`hdrout`	0x40000000	Header Transmitted
`pduin`	0x20000000	Packet Received
`pduout`	0x10000000	Packet Transmitted
`state`	0x04000000	State Change
`proc`	0x08000000	Procedural Trace
`error`	0x02000000	Error Situation
`logging`	0x01000000	Log Message

These values specify that the incoming or outgoing packets or frames (depending on which level is being traced). You can combine masks as a single number. For example, to trace both `pduin` and `pduout`, you would specify 0x30000000 (the logical or of 0x20000000 and 0x10000000).

This option may be abbreviated as `-tn`.

-traceoff

Disables tracing of subsystems specified with the `-entity` option. The trace file remains and you can format it to view the tracing messages.

This option may be abbreviated as `-tf`.

-log *class*

Controls the class of log messages that are enabled for the subsystems specified with the `-entity` option. The `nettl` daemon must be running when you issue this command.

This option may be abbreviated as `-l`.

class specifies the logging class. Available classes are:

Full	Abbrev	Mask
INFORMATIVE	I	1
WARNING	W	2
ERROR	E	4
DISASTER	D	8

You may specify *class* as a keyword or a numeric mask. The default logging class is DISASTER. The meaning for each possible *class* value is shown below.

`INFORMATIVE`	Describes significant operations and activities.
`WARNING`	Indicates abnormal events or conditions that have no permanent affect on subsystem integrity.
`ERROR`	Indicates abnormal events or conditions that have no permanent affect on subsystem integrity, but will cause a system call to fail and possibly an application program to terminate unexpectedly.
`DISASTER`	Signals an event or condition which WILL affect the overall subsystem or network operation and may cause several programs to fail or the entire card to shut down.

-fm [0|1|2] (for X.25 only)

Specifies the logging level for the X.25/9000 interface card's firmware at level 2 (frame level). **Note:** This option is valid for the series 800 only.

`0`	Logs disaster and error messages
`1`	Logs disaster, error, and warning messages.

2	Logs disaster, error, warning, and information messages.

This option may be abbreviated as `-fm`.

-card *dev_name*	Specifies the X.25/9000 interface card on which tracing and logging are to be initiated. This is the programmatic access name of the card, for example, `/dev/x25_0`. This option may only be used with the `-fm` or `-traceon` options. This option may be abbreviated as `-c`.

Note Only one X.25/9000 card may be traced at a time.

-file *filename*	Initializes tracing and creates a file with the *filename* specified and a suffix of `.TRCx` (where *x* = 0 or 1). All subsystems whose tracing is enabled with this command use this file. If `-file` is omitted, trace output will go to *stdout*. If the `-file` option is issued for a subsystem already being traced, the option is ignored unless that file is *stdout*. When tracing is enabled, every operation through that entity is recorded if it conforms to the *kind* mask. This option may only be used with the `-traceon` options. This option may be abbreviated as `-f`.
-size *limit*	Sets the trace buffer size (in kbytes). Trace messages will be held in the buffer until they are written to the file. Default value: 32 kbytes. Possible range: 1 to 512 kbytes. This option may only be used with the `-traceon` option. This option may be abbreviated as `-s`.
-m *byte*	Specifies the maximum number of bytes to trace. You may not need to capture the entire packet or frame. A number between 50 and 100 bytes is enough to capture the frame or packet header. The default is the entire packet or frame. This option may only be used with the `-traceon` option.

-tracemax *maxsize*

Specifies the maximum size of both trace files (TRC0 and TRC1) combined. *maxsize* stands for the number of kilobytes the combined size may be. The default size is 1000 kilobytes. The range is from 100 to 99999 kilobytes. If the trace buffer is not large enough to handle all incoming trace records, trace records can be lost. This option may only be used with the -traceon option.

-entity *subsystem [subsystem]*

Specifies the subsystem(s) to enable/disable for tracing or logging. The keyword "all" may be used to represent all configured subsystems.

The command *nettl -status* will display the names of all configured subsystems on the machine. See the "Logging and Tracing the OSI Stack" section earlier in this chapter for OSI subsystem names.

This option may only be used with the -traceon, -traceon, or -log options. The -entity option may be abbreviated as -e.

netfmt(1M) Syntax

netfmt formats common tracing and logging binary files.

Syntax

```
netfmt [-c config_file] [-f input file] [-p]
       [-t records][-F] [-l] [-v]
```

Options

-c *config_file*	Specifies the file that contains formatter filter configuration commands. The *config_file* must be in the search path, or be a complete pathname. If you omit `-c` and the file contains trace data, *netfmt* uses the `$HOME/.nettrc` file. If the file being formatted contains log data, *netfmt* uses the `$HOME/.netlogrc` file.
-f *input_file*	Specifies the file containing trace or log records recorded by *nettl*. If you don't specify `-f`, *netfmt* uses *stdin*. The file suffixes, `.LOGO`*X* or `.TRC`*X*, must be included in the *input_file* specification.
-p	Indicates *config_file* input is to be parsed. This allows you to perform a syntax check on the *config_file* specified with the `-c` option or the default file `$HOME/.nettrc` or `$HOME/.netlogrc`. The `-f` option is ignored. If the syntax is correct, *netfmt* terminates with no output or warnings.
-t *records*	Specifies the number of records to format from the end of the file. This allows you to quickly access the most recent information.
-F	Specifies that the input file is to be followed and not to be closed when end of file is encountered. *netfmt* keeps it open and continues to read from it as new data arrives. This is helpful when you want to watch events as they occur while troubleshooting a problem, or to record events to a hard copy device for auditing.

-l	Removes inverse video functions from the output stream. This option is useful if you are piping the output from *netfmt* to a non-video display device, such as a line printer.
-v	Causes *netfmt* internal debugging statements to be produced. This is helpful to HP field or support personnel in debugging *netfmt*.

Examples of nettl and netfmt Operation

Following are some examples of the *nettl* command.

Example 1

This example initializes the tracing/logging facility.

```
nettl -st
```

Example 2

This example changes log class to WARNING and DISASTER for all subsystems. DISASTER logging is always enabled for all subsystems.

```
nettl -l WARNING -e ALL
```

Example 3

This example turns on tracing for the TRANSPORT and SESSION subsystems (Input and Output headers only) and sends binary trace messages to files `/usr/adm/trace.TRC0` and `/usr/adm/trace.TRC1`.

```
nettl -tn hdrin hdrout -e transport session -f /usr/adm/trace
```

Example 4

This example determines trace status.

```
nettl -status TRACE
```

The resulting information should resemble the following:

```
     Trace Information:


  Trace File name:              stdout
  User's ID:            0       Buffer Size:     32768
  Messages Dropped:     0       Messages Queued:     0

  Subsystem Name:               Trace Mask:
  TRANSPORT                     c0000000
  SESSION                       c0000000
    .
    .
    .
```

Example 5

The following command reads the file `/usr/adm/trace.TRC0` for the binary data and uses `conf.file` as the filter configuration file.

```
netfmt -f /usr/adm/trace.TRC0 -c conf.file
```

Example 6

The following command formats the last 50 records in the file `/usr/adm/nettl.LOG00` (the default log file).

```
netfmt -f /usr/adm/nettl.LOG00 -t 50
```

Example 7

The following command uses the *follow* option (`-F`) and the configuration file to send disaster log messages (in the DISASTER.ONLY file) to the console.

```
netfmt -F -c DISASTER.ONLY < /usr/adm/nettl.LOG00 > /dev/console
```

```
DISASTER.ONLY contains

#                                -
# SUBSYSTEM     REQUEST_TYPE    ARGUMENT1     ARGUMENT2
#                                -

FORMATTER       filter          class           !*
FORMATTER       filter          class           DISASTER
```

Example 8

This example stops tracing/logging:

```
 nettl  -sp
```

The Formatting Filter Configuration File

This section describes the syntax and use of the *config_file* specified in the *netfmt* command with the `-c` option. If the `-c` option is not specified, one of the following default files will be used if it exists: `$HOME/.nettrc` for trace files or `$HOME/.netlogrc` for log files.

Specifying Filters

When *netfmt* begins operation it reads and interprets the *config_file* specified with the `-c` option or the default file `.nettrc` or `.netlogrc` in the home directory. The *config_file* specifies filters that will serve to reduce the number of trace or log records that will be formatted and written to *netfmt*'s *stdout* file. If no *config_file* can be found by *netfmt* all records are formatted.

netfmt reads the *config_file* from beginning to end. A filter enabled in the beginning of the file can be disabled in subsequent lines in the *config_file*. The filter types supported for all subsystems (referred to as "global" filters) are `class`, `kind`, `subsystem`, `log_instance`, `time_from`, `time_through`, `process_id`, `connection_id`, `path_id`, and `user_id`. There are also filters that only effect specific subsystems. An example of a subsystem filter is the `msgid` filter for the OTS, NETWORK, TRANSPORT, SESSION, and ACSE_PRES subsystems.

When *netfmt* reads a trace or log record, it compares the fields in the record to the filter settings specified in the *config_file*. If the record matches the filter settings, then the packet is formatted and written to *netfmt*'s *stdout* file. Otherwise, the packet is discarded.

Syntax for Filter Command Lines

Each command line specifies a criterion for selecting trace and log records from the input file.

General Format of the Filter Configuration File

netfmt reads the configuration file according to the following rules:

- The file is read from beginning to end.
- Data in the configuration file is interpreted a line at a time.
- A line beginning with a pound sign (#) is a comment. Comments are terminated by

a new line (end-of-line characters). All other characters appearing in a comment are ignored.

- Each filter command must appear on a separate line.
- White space such as spaces and tabs may be used freely to format filter command lines. A blank line is a valid construction.
- Keywords within a filter command line are case independent. For example, "error" is not distinguished from "ERROR".

Global Filters

```
FORMATTER FILTER  type [!] {value | *}
```

type	Is one of the following keywords: `class`, `kind`, `subsystem`, `log_instance`, `time_from`, `time_through`, `process_id`, `connection_id`, `path_id`, and `user_id`.
!	Negates the following value. Instead of selecting all records whose *value* matches the *value* specified in the filter command, the exclamation point matches all records whose *value* does not match the specified *value*.
value	Depends on the keyword specified for *type*.
*	Means all possible values. You can use it along with an exclamation point, "!*" to mean *not all* or *none*. This *value* is not valid for `time_from`, and `time_through`.

Global Filter Types

class	By default all log classes are formatted. To select only records of a single `class`, turn off all log classes with the `FORMATTER FILTER class !*` filter command, then specify one of the single classes listed below.

The possible *values* for the `class` type and their meanings are:

`INFORMATIVE`	Describes significant operations and activities.
`WARNING`	Indicates abnormal events or conditions that have no permanent degradation of the integrity of subsystems.
`ERROR`	Indicates abnormal events or conditions that have no permanent degradation of the integrity of subsystems, but will cause a system call to fail and possibly an application program to terminate.
`DISASTER`	Indicates an event or condition that WILL affect the overall subsystem or network operation and may cause several programs to fail or the entire subsystem to shut down.

kind

Trace type or mask type. A mask is a hexadecimal representation of a (set of) trace kind(s). You may enter either a keyword or mask is the *kind* value. The trace kinds and their corresponding masks are:

KEYWORD	**MASK**	**Meaning**
`hdrin`	0x80000000	Header Received
`hdrout`	0x40000000	Header Transmitted
`pduin`	0x20000000	Packet Received
`pduout`	0x10000000	Packet Transmitted
`state`	0x04000000	State Change
`proc`	0x08000000	Procedural Trace
`error`	0x02000000	Error Situation
`logging`	0x01000000	Log Message

subsystem	A complete list of subsystem names is available in the *nettl(1M)* HP-UX manual page. You can also list them out with the *nettl -status all* command. By using combinations of the operators below, it is possible to specify only a few subsystems to format out of a file containing many possible subsystems. No more than one subsystem name may be given per line; multiple lines will "OR" the request. You can turn off the subsystem name with the ! operator, given all subsystem except the one(s) indicated. Also, all subsystems may be specified with the * operator (default), or all subsystems turned off with the !* operator.
time_from	Starting time of the trace and log records to be formatted. Records whose time stamp comes before the specified time and date are not formatted. *value* has the following format: `hh:mm:ss.dddddd MM/DD/YY`, where the following meaning applies:

hh	stands for hours from 00 to 24.
mm	stands for minutes from 00 to 59.
ss	stands for seconds from 00 to 59.
.dddddd	stands for microseconds from 000000 to 999999.
MM	stands for months from 00 to 12.
DD	stands for days from 00 to 31.
YY	stands for years from 00 to 99.

time_through	Ending time of the trace and log records to be formatted. Records whose time stamp comes later than the specified time and date are not formatted. *value* has the following format: `hh:mm:ss.dddddd MM/DD/YY`. The syntax and semantics for this construction are described above.

OTS Subsystem Filters

```
OTS MSGID  [!] {message ID | *}
```

message ID	The Message ID Number that is part of each traced or logged message from each OTS subsystem. It can be recognized as the number contained in brackets in the second line of each traced or logged message. The Message ID Number is useful in determining the origin of a message. Refer to chapter 8, "Messages," for more details on Message ID Numbers.

Examples

The following examples show formatting filter commands in the configuration file.

Example 1

This example formatting filter command file instructs *netfmt* to format only INFORMATIVE log messages coming from the ns_ls_x25 subsystem that occurred from 10:31:53 to 10:41:00 on November 23, 1990.

```
#                          REQUEST_TYPE     ARGUMENTS
#
FORMATTER FILTER           time_from        10:31:53   11/23/90
FORMATTER FILTER           time_through     10:41:00   11/23/90
FORMATTER FILTER           class            !*
FORMATTER FILTER           class            INFORMATIVE
FORMATTER FILTER           subsystem        !*
FORMATTER FILTER           subsystem        ns_ls_x25
```

Example 2

This example formatting command file instructs *netfmt* to format only hdrin kind coming from the SESSION subsystem.

```
#                          REQUEST_TYPE  ARGUMENTS
FORMATTER FILTER           kind          !*
FORMATTER FILTER           kind          hdrin
FORMATTER FILTER           subsystem     !*
FORMATTER FILTER           subsystem     SESSION
```

Example 3

This example formatting command file instructs *netfmt* to format only those messages containing message ID 9919 in trace kind `hdrin` or `hdrout` for subsystem ACSE_PRES.

```
#                             REQUEST_TYPE  ARGUMENTS
FORMATTER FILTER              kind          !*
FORMATTER FILTER              kind          hdrin
FORMATTER FILTER              kind          hdrout
FORMATTER FILTER              subsystem     !*
FORMATTER FILTER              subsystem     ACSE_PRES
OTS                           msgid         !*
OTS                           msgid         9919
```

General Format of the nettlgen.conf File

The Network Tracing and Logging Configuration File (`/etc/conf/nettlgen.conf`) specifies the initial configuration of the network tracing and logging facility. This configuration information includes:

- which network subsystems will be logged
- the default log level for each subsystem
- the log file name and size
- whether disaster messages shall be logged to the system console

Most system administrators have no need to modify this file. However, under certain circumstances the default values for log files, their sizes and console logging may need to be changed. To change these default values, edit the `/etc/conf/nettlgen.conf` file using a text editor.

After the `/etc/conf/nettlgen.conf` file has been edited the *nettlgen* command must be used to implement the changes. The *nettlgen* command reads the configuration file and configures the network tracing and logging facility accordingly.

Syntax of the nettlgen.conf File

The `/etc/conf/nettlgen.conf` file contains logging information records and subsystem information records. The *nettlgen* command reads the configuration file and configures the network tracing and logging facility accordingly. The *nettlgen* command expects the `/etc/conf/nettlgen.conf` file to adhere to the following syntax rules:

- The file is read from beginning to end.
- Data in the configuration file is interpreted one record at a time.
- A record beginning with a pound sign (#) is a comment. Comments are terminated by a new line (end-of-line characters). All other characters appearing in a comment are ignored.
- Each configuration command record must appear on a separate line.
- Each field in the record is separated by a single colon (:).
- No backslash characters (\) are allowed in any line.
- No whitespace characters are allowed in any record field except within the group name field.

- The word "NULL" may be used to represent an empty field.
- If more than one LOG record is encountered, only the last LOG record in the configuration file is used.

```
LOG  console_flag port_size file_size filename
SS   ID mnemonic default_class library catalog function
     options group
```

Configuration Record Parameters

console_flag	Defines whether *nettl* will perform console logging or not. 0 indicates no console logging, and 1 indicates console logging will be used.
port_size	Specifies the size of the internal log message buffers for each subsystem. The value is specified in 1Kbyte units (that is, 8 stands for 8192 bytes). Eight is the recommended value.
file_size	Specifies the maximum size of both log files combined. That is each log file will not exceed one half of the specified *file_size*. The value is specified in 1Kbyte units (that is, 1000 stands for 1 megabyte). One thousand is the recommended value.
filename	Indicates the path and file name of the log file. *nettl* shall add `.LOG00` or `.LOG01` to this name. The default *filename* is `/usr/adm/nettl`.

Note The remaining parameters are for the subsystem records. HP does not recommend changing these fields or adding new records without direct supervision of an HP representative.

ID	Identifies the subsystem with a number between 0 and 127 set at the factory.
mnemonic	Identifies the subsystem with a character string set at the factory.
default_class	Identifies the default logging class for the subsystem. This value is set at the factory.

library	Identifies the library file containing the formatting functions for the subsystem. This value is set at the factory.
catalog	Identifies the message catalog for the subsystem. This value is set at the factory.
function	Identifies the function to format trace and log data for the subsystem. This value is set at the factory.
options	Identifies the function to get options for the subsystem. This value is set at the factory.
group	Identifies the group for the subsystem. This value is set at the factory.

Example

This example shows the LOG record with the default setting: console logging is enabled, 8192 bytes are reserved for each logging port, 1 megabyte of file space is used for the log files, and the log file pathnames are `/usr/adm/nettl.LOG00` and `/usr/adm/nettl.LOG01`.

```
LOG:1:8:1000:/usr/adm/nettl
```

7

OSIDIAG

A detailed description of the OSI Diagnostic (osidiag) program

Introduction

OSIDIAG provides a convenient and consistent means to verify that the OSI services and stack provided by your HP system (as well the remote node) are operational.

Tests are available for FTAM, MMS, X.400, CMIS, APRI, Session, Transport, X.25 and LAN.

When Is OSIDIAG Used

This tool would be used in the following situations:

- **Installation** - when the product is first installed or updates are installed, you can easily generate network activity to expose problems in SW or HW installation or configuration.
- **Network Reconfiguration** - you will be able to generate network activity to verify or expose problems in the new configuration.
- **Application Debugging** - when developing applications, osidiag can act as a "sanity" check to determine if problems encountered are application specific or global to the network.
- **Support** - if a problem is encountered which is system wide, it will be easier for HP support to interpret the problem if it can be expressed in terms of a failure through osidiag rather than in terms of the user's application.
- **Demonstration Purposes** - osidiag can be useful for demonstrating some of the products features to persons unfamiliar with the product.

Other OSIDIAG Features

Other important features of OSIDIAG are the following:

- **Parameter Files** - OSIDIAG allows you to read test parameter values from a file. This makes repeated tests against the same node simpler.
- **Result Files** - OSIDIAG allows a record of the tests to be written off to a file in addition to being displayed on your screen. These files should contain sufficient information for your HP support representative in the event you encounter difficulties.
- **Default Parameter Values** - OSIDIAG automatically reads your configuration files to supply default values for most parameters. This makes running local verification very easy.
- **Integrated Tracing and Logging** - OSIDIAG's tests are integrated with the tracing and logging facilities for the various components of the OSI stack. By enabling the trace facilities provided through OSIDIAG, you have the maximum diagnostic capabilities through a single tool.

Command Line

The basics of executing osidiag are described below. The subsequent sections will provide more detail about the specific tests available for each service, the parameters you can use to control the test behavior and how to use the various utilities.

Syntax

```
/etc/osidiag [-e subsystem [-t test_mnemonic]] [-p parm_file] [-d dest_addr]
[-w wait_time] [-n] [-h]
```

Description

osidiag is used to test the ability to communicate with a remote OSI system using the FTAM, MMS, X.400, CMIS, APRI, Session, Transport, X.25, FDDI and 802.3 LAN services. The test executed may be specified on the command line or if the test is not specified a menu based interactive interface will be presented. Some features of osidiag (i.e. tracing and logging) are only available through the interactive interface.

osidiag allows the user to select some of the parameters used on the service calls (for example the address of the remote entity, or data to be sent). These parameters may be preset by storing them in a parameter file or the caller may opt to be prompted by the interactive interface for their values. Most tests will run in loopback if the default parameters are used.

-e subsystem
: Specifies which subsystem is to be tested. The values supported are **FTAM, MMS, X400, CMIS, APLI, ROSE, TRANSPORT, SESSION, X25,** and **LAN.** If this parameter is specified then the caller will be taken to the test screen for the subystem selected. If no value is specified the user be placed at osidiag's main menu which allows testing of all supported layers. This option must be selected in order to select a test case from the command line (see -t).

-t test_mnemonic
: Specifies a particular test case is to be executed. When this option is selected the interactive interface is *not* used. Tests available at most layers are **loop** for a loopback test, **srv** for a server test and **con** for a connect test. Specifying a test case name of "h" will produce a list of valid test names for the specified layer.

-p parm_file
: Indicates that the file *parm_file* contains a list of parameters to be used to override default values provided by osidiag. Note that new parameter files can be read into osidiag through the utility section of the interactive interface.

-d dest_addr
: Specifies the address of the remote to be tested with. This can also be specified through a parameter file or through the interactive interface. If the destination address is also specified in a parameter file, the parameter file value will override this value.

-w wait_time

Specifies the time (in seconds) which osidiag will wait for a response from the remote. This applies primarily to server tests. The default time is 30 seconds. If this value is also given in a parameter file, the parameter file will override the value on the command line.

-n

The "no more" flag. If present the output of the test case will not be piped through the UNIX utility /bin/more. By default the test output is piped through more to prevent it from scrolling off the screen before being read.

-h

The "hex-only" flag. If present parameters which are hexadecimal values will always be displayed in hex. By default osidiag will display hexadecimal strings which are comprised soley of printable ASCII characters in ASCII (e.g. the hex string 68656C6C6F would be displayed as "hello").

Examples

```
osidiag
```

In the example above the interactive interface will be presented. From that point the user may select the entity to test and the test to be performed.

```
osidiag -e TRANSPORT
```

In the example above the interactive interface will be presented. However only the tests available for the Transport layer will be displayed.

```
osidiag -e SESSION -t loop
```

In the example above the the Session layer loopback test will be executed. The default parameters extracted from the configuration file will be used to determine addressing information.

```
osidiag -e FTAM -t con -d 0001.0001.0001.6873443332
```

In the example above the the FTAM connection test will be executed. The FTAM responder with the address "0001.0001.0001.6873443332" will receive the connection request.

```
osidiag -e X25 -t srv
```

In the example above the the X.25 server test will be executed. This will result in our awaiting a connection indication at the configured X.121 address and subaddress.

```
osidiag -p /tmp/ibm_parms
```

This will bring up the standard osidiag interactive interface. However the parameters specified in the file "/tmp/ibm_parms" will override the default values.

Interface

Osidiag provides two means to execute test cases:

- Through an interactive menu interface.
- By specifying the test case to be performed at the command line.

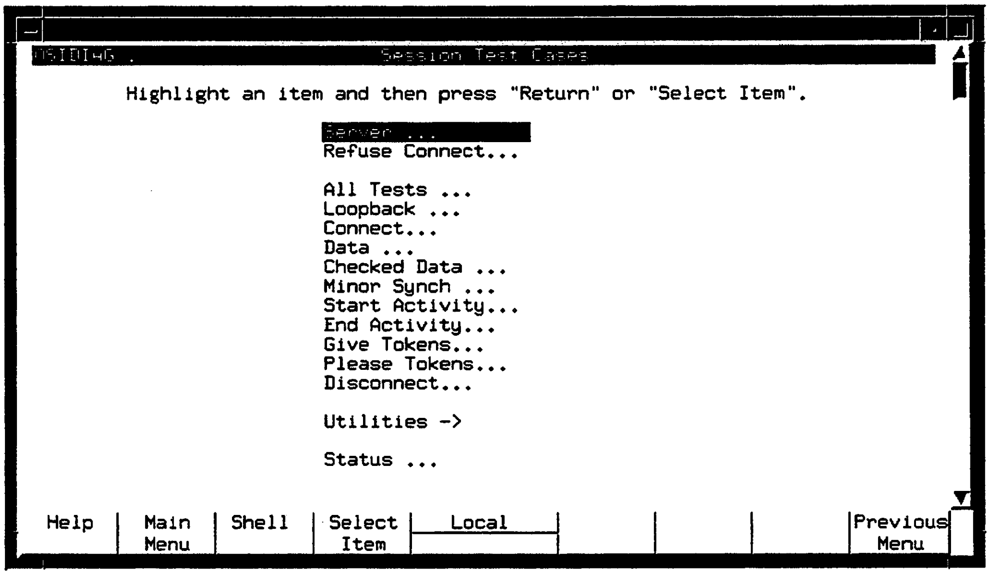
The interactive interface will be brought up if you simply enter "/etc/osidiag".

If you wish to run a test without the interface you must specify both the "-e" and the "-t" command line options.

Regardless of the interface used the test case output will look the same.

Menus

A typical test case menu given when the interactive interface is used is shown below:



```
OSIDIAG .                    Session Test Cases

       Highlight an item and then press "Return" or "Select Item".

                          Server ...
                          Refuse Connect...

                          All Tests ...
                          Loopback ...
                          Connect...
                          Data ...
                          Checked Data ...
                          Minor Synch ...
                          Start Activity...
                          End Activity...
                          Give Tokens...
                          Please Tokens...
                          Disconnect...

                          Utilities ->

                          Status ...

  Help  | Main | Shell | Select |   Local   |   |   |Previous
        | Menu |       |  Item  |           |   |   |  Menu
```

One test case name will appear in inverse video, this is the currently selected test. In order to change the selection, press the TAB key. When the test case you want is displayed press the **Select Item** function key or simply press <return>.

The **Main Menu** key will return you to the topmost menu. Typically this will have the same affect as the **Previous Menu** key since osidiag does not have a very deep menu tree.

The **Help** key will give you information about the menu item which is currently highlighted.

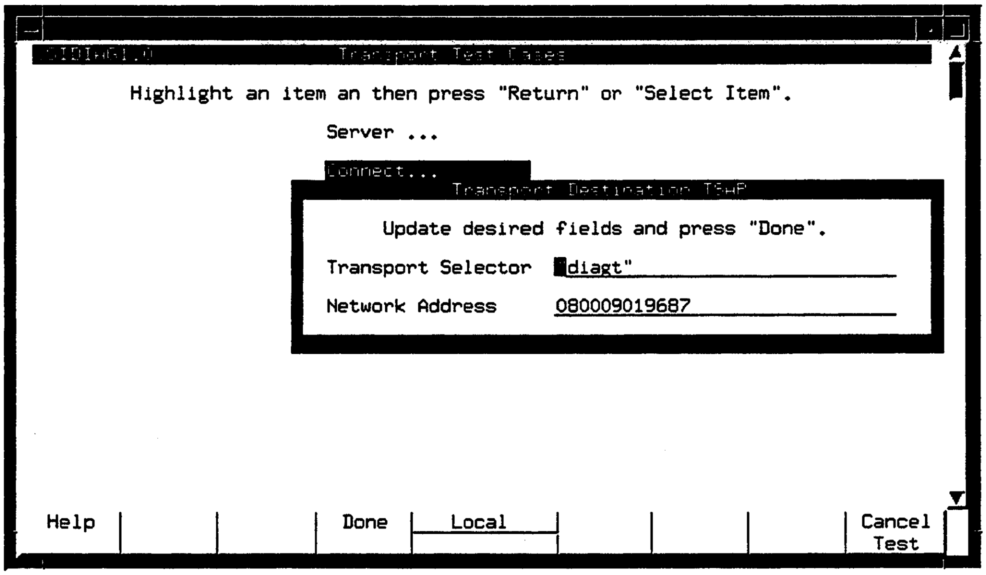
The **Shell** key allows you to temporarily leave osidiag and get a shell. Return to osidiag by typing "exit" at the shell prompt.

All the menus allow you to go to the **Utilities** section of osidiag. The utilities allow you to do things like access parameter files, create result files, or enable tracing and logging.

All the layers provide a **Status** operation. This will indicate whether the layer is up and display some configuration and statistical information particular to that layer.

Pop-up Parameter Windows

A typical parameter window is shown below:



```
OSIDIAG1.0                    Transport Test Cases

        Highlight an item an then press "Return" or "Select Item".

                        Server ...

                        Connect...
                              Transport Destination TSAP

                            Update desired fields and press "Done".

                        Transport Selector  "diagt"

                        Network Address     080009019687
```

These pop up windows are displayed after you select an osidiag operation (e.g. Transport Loopback test). The underlined fields require input. If there are multiple parameters on a window use the TAB key to move to the various fields. Descriptions of the parameters requested by osidiag are given in the sections describing each layers tests. Information about the parameters may also be gotten by pressing the **Help** key.

The **Done** key will accept the parameter values shown and proceed with the test or to the next parameter window.

The **Cancel Test** key will abort the test case and return you to the test case menu.

Osidiag Output

The output of a typical osidiag test case is shown below:

```
entity:    TRANSPORT
test case: Connection
------------------------------------
tp_my_tsap          : .."diagt"."hpindka"
tp_dst_tsap         : "diagt"."hpindka"
tp_class            : 4
tp_alt_class        : N
tp_cdata_val        :
tp_cdata_file       :
tp_ddata_val        :
tp_ddata_file       :
------------------------------------
t_open()               fd: 07   result: 0  t_errno: 0
t_bind()               fd: 07   result: 0  t_errno: 0
t_connect()            fd: 07   result: 0  t_errno: 0
t_look()               fd: 07   result: 0  t_errno: 0
                         indication:  T_CONNECT
t_rcvconnect()         fd: 07   result: 0  t_errno: 0
t_snddis()             fd: 07   result: 0  t_errno: 0
t_unbind()             fd: 07   result: 0  t_errno: 0
t_close()              fd: 07   result: 0  t_errno: 0
------------------------------------
class:          4
flowctrl:       -1
------------------------------------
test status:  PASSED
=========================================================================
```

The output is divided into the following five sections by hyphens "---":

- Test Case Description
- Test Parameters
- Primitive Operations
- Returned Information (optional depending on test case)
- Test Status

The test case description tells which component is being tested and which test case was chosen.

The test parameter description displays the parameter values which osidiag uses for this test case. This can be helpful in the event of a typo. This is also useful when generating

"Result Files" to give complete information about the test case being run.

Note that some of the parameters displayed will not be prompted for (by default). An example above is the tp_cdata_val parameter. The reason is that the default value for this parameter is sufficient in most cases (in this case NULL). The subsection "Set Prompt Level" under "Utilities" describes how you can alter the default prompting.

The primitive operation summary shows the name of the primitive operation being performed, the channel number (or equivalent) being used and the result code returned for that operation. This is helpful in following the steps osidiag is performing to execute the test case.

The returned information is that sent back to us from the remote. In this example the information was carried on the connection confirmation. It shows the variaous services and parameter values negotiated for the connection. Some test cases (e.g. FTAM File Delete) don't report any information so this section is omitted.

The test status indicates if the test PASSED or FAILED. See the "Troubleshooting" chapter in this manual for more information.

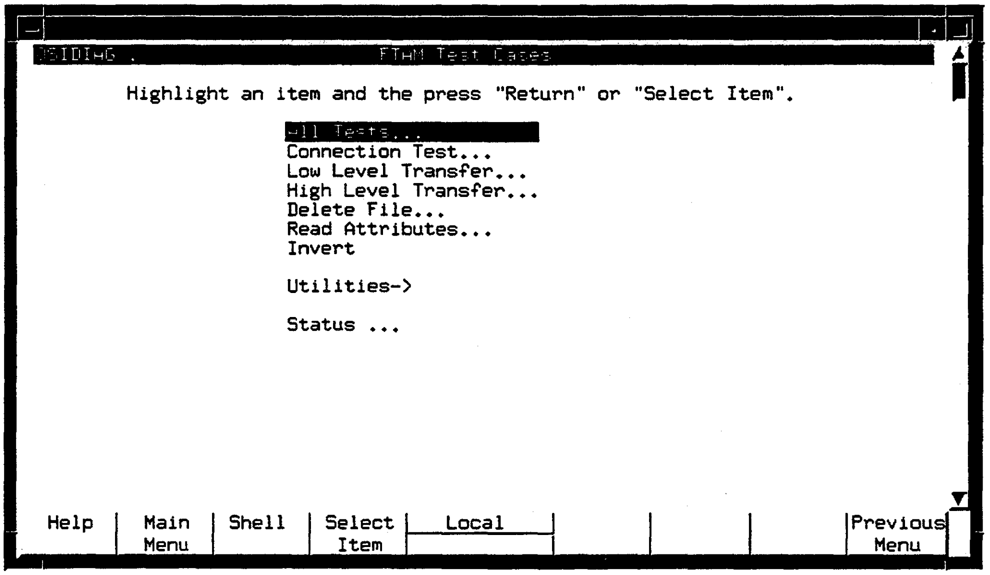
FTAM Tests

Osidiag allows you to verify that HP's FTAM initiator can connect to the FTAM responder of a remote. You can also transfer files, delete files, and view their attributes.

These operations can also be performed interactively through the ftam(1) command. The ftam(1) command should be used for routine FTAM operations. However when troubleshooting, osidiag's ability to automate tracing, report operation errors, and create result files will make osidiag more useful.

Test Menu

The test case menu displayed for the FTAM tests is shown below:



```
OSIDIAG .                      FTAM Test Cases

        Highlight an item and the press "Return" or "Select Item".

                      All Tests...
                      Connection Test...
                      Low Level Transfer...
                      High Level Transfer...
                      Delete File...
                      Read Attributes...
                      Invert

                      Utilities->

                      Status ...
```

All Tests

This will run the connection test, the high and low level transfer, the read attributes and the delete file test. Upon completion, the file you specify will have made a round trip transfer, and can be diff(1)'d with the original to ensure file integrity.

Connect

The connect test will result in a connection request being sent to the remote FTAM responder peer. After a confirmation is received from the remote the connection is dissolved.

Low Level Transfer

This test uses the low-level FTAM primitives to perform a file transfer. This is in contrast to the high-level FTAM operation ft_fcopy() which performs the transfer with one call. This test provides more detail about which portion of a file transfer is failing if problems occur.

High Level Transfer

This test uses the high-level FTAM function ft_fcopy() to transfer a file. NOTE that the high level transfer only supports the use of DDN's or aliases. The low level and connection tests, allow you to specify Presentation addresses.

Delete File

This test will attempt to delete the specified file from the specified remote node.

Read Attributes

This test will attempt to read attribute information from a remote file.

Invert

This will switch the source and destination parameters. This is useful when you are performing a test of writing a file out and reading it back from the same remote. It is invoked automatically as part of the All Tests operation.

Utilitites

This will take you to the Utilities menu.

Status

This will invoke the program osistat to display current application layer status and statistics.

FTAM Model of Interaction

When performing a file transfer through FTAM there may be three FTAM processes involved: the **initiator**, the **source responder** and the **destination responder**.

The initiator process always resides on the local node. Typically either the source or destination responder will also reside on the local node and the other responder will reside on a remote node. It is legal however to transfer a file between two remote responders.

In order to "write" a file from the local to the remote node we would specify the source as being the local node and the destination as being the remote.

In order to "read" a file from the remote we would specifiy the source as being the remote node and the destination as being the local.

If you specify a NULL DDN for the "High Level Transfer", "Delete File" or "Read Attributes" test it will mean that the local responder will be used.

Normally to increase performance the local initiator process will not use FTAM to communicate with the local responder, but will rather directly access the Virtual Filestore (VFS) of the local node.

You must still perform the operations to "connect" to the responder, however no real PDU traffic is generated when the responder is local.

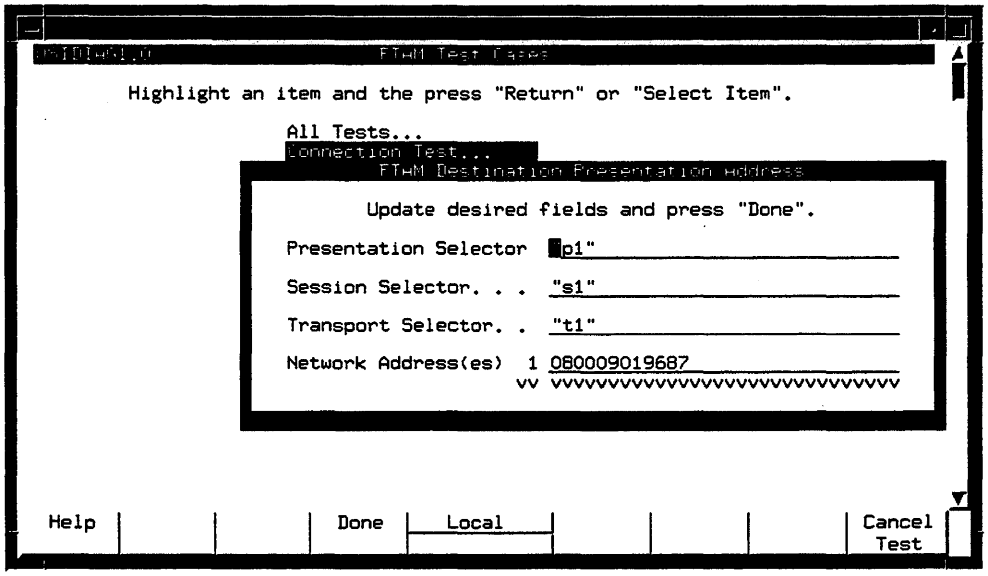
This behavior can be overridden through the environment variable "USE_VFS_LINK". When set to the value "no", FTAM will use the network to communicate with local responder. Osidiag will automatically set this environment variable to "no" before running the test unless you have already set it. This guarantees that the network is exercised for the test.

Parameters

The following describe the parameters which the various FTAM tests require.

Destination P-Address

The data entry window presented by osidiag in order to get the address of the remote is shown below:



```
OSIDIAG1.0                 FTAM Test Cases

        Highlight an item and the press "Return" or "Select Item".

                       All Tests...
                       Connection Test...
                          FTAM Destination Presentation Address

                          Update desired fields and press "Done".

                       Presentation Selector  "p1"
                       Session Selector. . .  "s1"
                       Transport Selector. .  "t1"
                       Network Address(es)  1 080009019687
                                           vv vvvvvvvvvvvvvvvvvvvvvvvvvvvvvv
```

The Presentation address of the remote is composed of four components: the **presentation selector** the **session selector** the **transport selector** and the **network address**. These fields should contain the Presentation address which is configured for the remote FTAM Responder.

The default value presented by osidiag is that configured for our local FTAM responder.

Note that if the selector values are all ASCII that you may enter them as quoted strings. Otherwise specify them in hex.

You can determine the network address of a remote HP node by running osiconf on that node and viewing the subnetwork configuration.

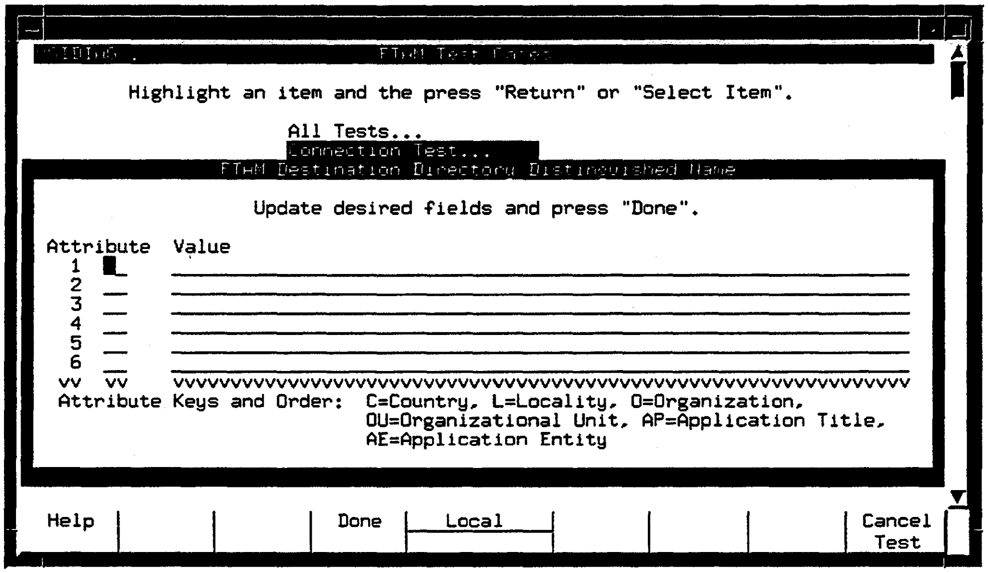
You can determine the P,S and T selector values by running osiconf and viewing the FTAM configuration for the responder.

Responder DDN's

For the delete file and read attribute tests you will be prompted for a single responder. For the high level transfer you will be prompted for a source and destination responder.

A Directory Distinguished Name (DDN) is also refered to as an Application Entity Title (AE Title). It is a structured name composed of well defined attributes and their values.

The window presented looks as follows:



```
OSIDIAG .                    FTAM Test Cases

        Highlight an item and the press "Return" or "Select Item".

                         All Tests...
                         Connection Test...
                 FTAM Destination Directory Distinguished Name

                      Update desired fields and press "Done".

 Attribute  Value
    1   _   ____________________________________________________________
    2   _   ____________________________________________________________
    3   _   ____________________________________________________________
    4   _   ____________________________________________________________
    5   _   ____________________________________________________________
    6   _   ____________________________________________________________
   vv  vv   vvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvv
  Attribute Keys and Order:  C=Country, L=Locality, O=Organization,
                             OU=Organizational Unit, AP=Application Title,
                             AE=Application Entity
```

The column of fields labeled "Attribute" should contain the appropriate attribute key (as listed at the bottom of the form). The column labeled "Value" should contain the attribute value.

As an example you might enter:

```
O    hpindzz
AP   ftam
AE   resp
```

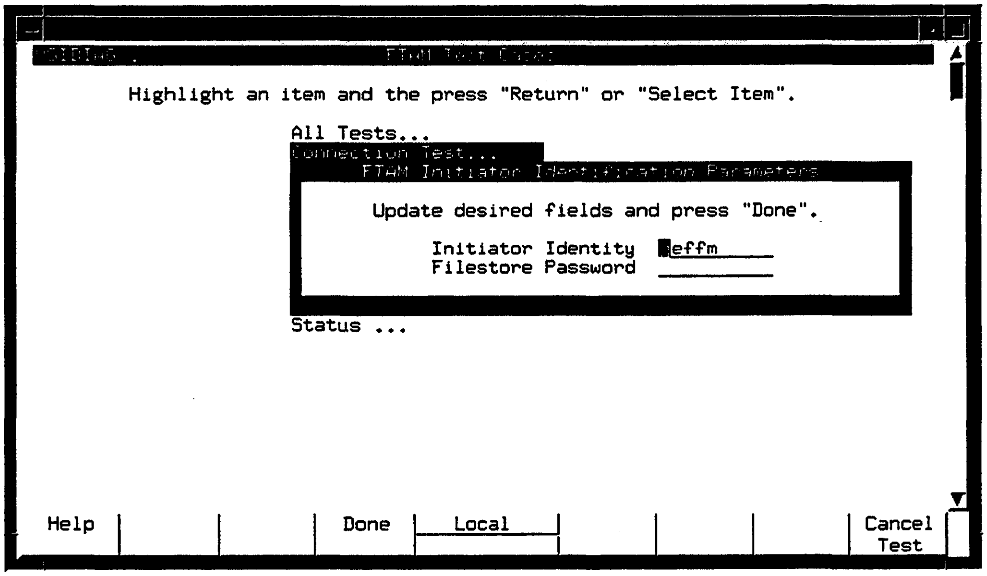
This indicates that the responder's DDN has three components, an Orginization Name, an Application Title and an Application Entity.

To determine alternate DDN' you should examine the configured names for remote applications as configured under osiconf(1M). These can be displayed by selecting "Status..." from the test menu. Status invokes osiconfchk(1M) in addition to osistat, to produce a summary list of configured values. The values configured in the file "remote_app" give the AE Title and their corresponding Presentation Address.

FTAM User Id

The FTAM User Id information corresponds to the HP-UX login information for HP nodes. It is used to restrict access to the file system to authorized users.

The window for this information is as follows:



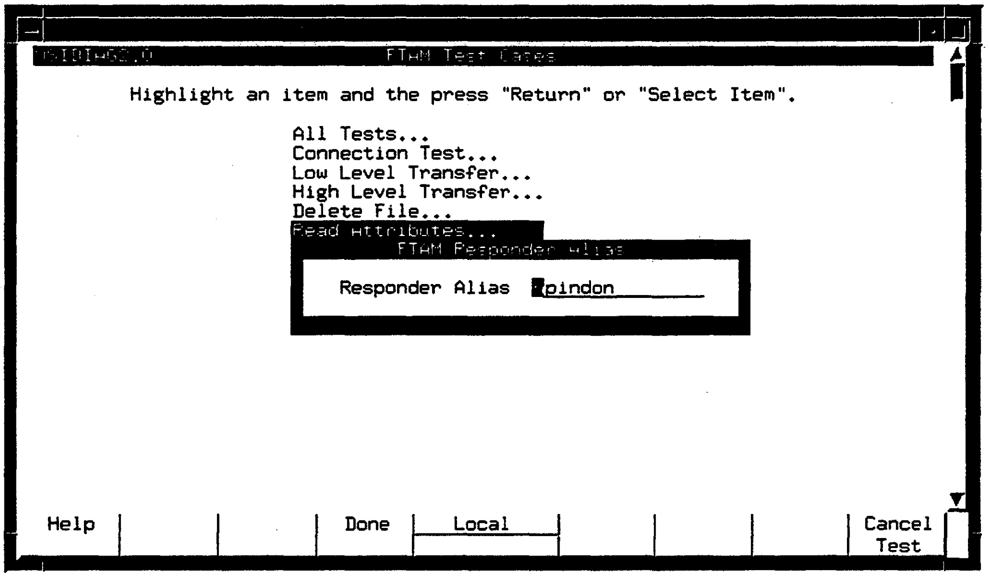
```
OSIDIAG .                          FTAM Test Cases

        Highlight an item and the press "Return" or "Select Item".

                        All Tests...
                        Connection Test...
                              FTAM Initiator Identification Parameters

                            Update desired fields and press "Done".

                                Initiator Identity  effm
                                Filestore Password

                        Status ...

  Help  |     |     | Done | Local |     |     | Cancel
                                               | Test
```

The field **Initiator Identity** should contain your login name (or equivalent for other vendor's systems), and the **Filestore Password** should contain the password associated with the login. Note that the password will not be displayed as you type it.

FTAM Alias

The FTAM Alias is an easy way to identify a remote FTAM responder process. It is a single name, which is expanded into a full Directory Distinguished Name using the substitution rule called "ftam_ddn_lookup_path" given in the configuration file "/etc/net/osi/conf/local_app".

The window for this information is as follows:



```
OSIDIAG2.0                    FTAM Test Cases

        Highlight an item and the press "Return" or "Select Item".

                          All Tests...
                          Connection Test...
                          Low Level Transfer...
                          High Level Transfer...
                          Delete File...
                          Read Attributes...
                                FTAM Responder Alias

                             Responder Alias  epindon
```

Example

The following shows a successful FTAM connection test.

```
entity:    FTAM
test case: Connection
-----------------------------------
ft_my_ddn           : /C=us/O=hp/AP=ftam/AE=ftam_init
ft_dst_ident        : root
ft_dst_fst_pwd      :
ft_dst_paddr        : "p1"."s1"."t1"."hpindme"
-----------------------------------
ft_aeactivation()     ae_label: 0x7484000b   result: 0
ft_connect()          conn_id:  0x0000ffff   result: 0
ft_rrequest()         conn_id:  0x0000ffff   result: 0
ft_aedeactivation()   ae_label: 0x7484000b   result: 0
-----------------------------------
service class:
                     ACCESS
functional units:
                     READ
                     WRITE
                     FILE_ACCESS
                     LTD_FILE_MGMT
                     ENH_FILE_MGMT
                     GROUPING
attribute groups:
                     STORAGE
                     SECURITY
contents types:
                     FTAM_3
                     FTAM_2
                     FTAM_1
-----------------------------------
test status:  PASSED
=======================================================================
```

The first two lines indicate the test being executed. The next section of output is the parameter display. The third section is the dialog display. The fourth section shows the Negotiated Parameters. Lastly the test status is shown.

Parameter Display

The parameters used for the connection test are displayed after the test case name. The meaning of all available FTAM parameters are described in the table later in this section.

Dialog Display

The following FTAM primitives are executed to perform a connection test. More information about these operations can be gotten from the FTAM Programmer's Guide or the online manual pages.

- **ft_aeactivation()** - this creates a "Server Provider Process" (see "Architecture" diagram at beginning of "Concepts" section). The SPP is responsible for the actual encoding and decoding of FTAM PDU's it also interacts with the stack. This operation does not result in any network activity.
- **ft_connect()** - this operation sends an FTAM connect PDU and awaits a connect response.
- **ft_rrequest()** - this sends an FTAM release request PDU and awaits the release response.
- **ft_aedeactivation()** - this call terminates the SPP. It does not result in any network activity.

Negotiated Parameters

At the completion of the FTAM connect test, the parmaters negotiated with the remote responder are displayed.

- **service class** This is a higher level grouping of the functional units. The possible values are UNCONSTRAINED which allows all functional units; MANAGEMENT which disallows file access, read and write; TRANSFER which disallows file access; TRANSFER_AND_MGMT which disallows file access and requires management and ACCESS which requires read, write and file access and permits management.
- **functional units** This indicates what primitives are supported by the server. Each name corresponds to a group of primitives. READ and WRITE are self explanatory, FILE_ACCESS indicates the ability to extract specific FADUs from a file. LTD_FILE_MGMT allows files to be created and deleted as well as attributes of files to be read. ENH_FILE_MGMT allows file attributes to be modified. GROUPING indicates that primitives can be sent as groups (i.e. select and open carried on one PDU), this is required by NBS.
- **attribute groups** This indicates what file attributes may be accessed from the remote. The KERNEL attributes are always supported and are not displayed. PRIVATE allows the private attribute to be accessed. STORAGE group contains information on the dates the file was modified and who did it. The SECURITY group contains info on the access control and qualifications.
- **contents type** This indicates the types of FTAM files which are supported by the remote. In oversimplified terms: FTAM-1 is an unstructured text file, FTAM-2 is a structured text file and FTAM-3 is an unstructured binary file.

Read Attributes Display

The following shows a successful read attributes test.

```
entity:    FTAM
test case: Read Attributes
------------------------------------
ft_my_ddn           : /C=us/O=hp/AP=ftam/AE=ftam_init
ft_dst_ident        : jeffm
ft_dst_fst_pwd      :
ft_dst_ddn          : /C=us/O=hp/AP=ftam/AE=ftam_resp
ft_dst_fname        : /etc/motd
------------------------------------
ft_aeactivation()      ae_label: 0x748a000b   result: 0
ft_frattributes()      ae_label: 0x748a000b   result: 0
ft_aedeactivation()    ae_label: 0x748a000b   result: 0
------------------------------------
filename             : /etc/motd
permitted actions    :
                        READ
                        REPLACE
                        EXTEND
                        ERASE
                        READ ATTRIBUTE
                        CHNAGE ATTRIBUTE
                        DELETE FILE
contents type        : FTAM_1
storage account      : no value available
create date/time     : 00:00:0000000  000000
mod date/time        : 18:00:00Z  870921
read date/time       : 17:16:42Z  900604
att mod date/time    : 00:00:0000000  000000
id of creator        : root
id of modifier       : no value available
id of reader         : no value available
id of att mod        : no value available
file availability    : IMMEDIATE
access control       :
                         user             R----ACD
                         group            R----ACD
                         other            R----ACD
filesize             : 40
future filesize      : 0
legal qual           : no value available
private use          : (NOT PRESENT)
------------------------------------
test status:  PASSED
========================================================================
```

Attribute Information

After completing a successful read attributes operation the following values are reported.

- **filename** this is the name of the file.
- **permitted actions** This indicates what can be done against the file: the possibilities are READ read the file; INSERT add a FADU; REPLACE replace a FADU; EXTEND add to the file; ERASE remove the contents of the file; READ ATTRIBUTES read this data; CHANGE ATTRIBUTES modify the attributes; DELETE FILE to remove from filestore; TRAVERSAL ability to walk through FADUS; REVERSE TRAVERSAL and RANDOM ORDER are not compliant with NBS but may be reported.
- **contents type** - Indicates whether the file is FTAM-1, 2 or 3.
- **storage account** - Gives the name of an authority to be charged.
- **time of creation** - Time and date file was created.
- **time of last mod** - Time and date file was modified.
- **time of last read**
- - Time and date file was read. **time of last attr mod** - Time and date attributes were modified.
- **identity of creator** - self explanatory.
- **identity of last modifier** - self explanatory.
- **identity of last reader** - self explanatory.
- **identity of last attr mod** - self explanatory.
- **file availability** - May be either IMMEDIATE or DEFERRED indicating when it can be accessed.
- **filesize** - Size of file in octets.
- **future filesize** - Maximum size file can be grown. Ignored by HP responders.
- **access control** - list of filestore users and their permissions. The permissions are the same as those given by the fls(1) command. The form is "RIPXEACD", if all permissions are available. A "-" in place of a permission indicates it is not available. These stand for (R)ead, (I)nsert, re(P)lace, e(X)tend, (E)rase, read (A)ttribute, (C)hange attribute and (D)elete file.
- **legal qualifications** - not used by HP.
- **private use** - not used by HP.

Interpreting Errors

If an FTAM test operation fails, the following output will appear in the status section at the end of the output:

```
-----------------------------------
test status:  FAILED
operation:    ft_connect()
return code:  35
vendor code:  3000
diagnostic:
  further details:  Incorrect filestore password for user 'freddy'
  error id:         2000
  error observer:   FT_RESPONDING_FILE_SERVICE_USER
  error source:     FT_INITIATING_FILE_SERVICE_USER

(FTE035) Confirmation failed
(FTV000) No additional information available
Log Instance:  52

======================================================================
```

The meaning of each field is as follows:

- **test status** - indicates whether we passed or failed. Any error encountered during the operation (including setting up tracing and logging, if requested) will result in a status of failed.
- **operation** - indicates at what point in the test we encountered the first error. Typically this will be the name of an FTAM operation.
- **return code** - this is the return code from the first error encountered. The meaning of this error code is displayed below the "diagnostic:" fields. In this case it is "(FTE035) Confirmation Failed".
- **vendor code** - this field may contain additional information about the error. Any text associated with it will be displayed immediately after the return code text. In this case it is "No additional information available".
- **diagnostic** - these fields contain additional FTAM error information. The **further details** may contain more specific information about the error. The **error id** field contains an implementation specific error number. For HP the "further details" will contain more specific information than the "error id". The **error observer** and **error source** indicate what part of the FTAM machine detected the error and what part it believes caused the error.
- **Formatted error text** - after the above fields are displayed, osidiag will provide any textual error information which is available. If there is a "log instance" given, then this may be matched to an error in the nettl log file.

Further Information

Further information on interpreting errors may be found in the Troubleshooting section of this manual. Also consult the "FTAM Reference Manual."

Parameter Summary

The following table summarizes the FTAM parameters which are configurable by the caller of osidiag.

FTAM Parameters

parameter	syntax	description
ft_src_paddr	paddr	specifies the presentation address of the source of a transfer. Not required if a ddn is given.
ft_dst_paddr	paddr	specifies the presentation address of the destination of a transfer. Not required if a ddn is given.
ft_my_ddn	ddn	specifies the directory distinguished name of the local FTAM Initiator AE.
ft_src_ddn	ddn	specifies the ddn of the FTAM Responder which provides the source file on a transfer.
ft_dst_ddn	ddn	specifies the ddn of the FTAM Responder which is the recipient of the file transfer. Also used for non-transfer calls (e.g. delete).
ft_dst_alias	string	specifies the alias of the FTAM Responder which is the recipient of the file transfer. Also used for non-transfer calls (e.g. delete).
ft_src_alias	string	specifies the alias of the FTAM Responder which provides the source file on a transfer.
ft_dst_ident	string	specifies the identity of initiator value to be carried on the ft_initialize (connect) request for the recipient of the file transfer. Also used for non-transfer calls (e.g. delete).
ft_src_ident	string	specifies the identity of initiator value to be used when connecting to the file store containing the source for the file transfer.
ft_dst_fst_pwd	string	specifies the filestore password used when connecting to the destination filestore. Also used for non-transfer calls (e.g. delete).
ft_src_fst_pwd	string	specifies the filestore password used when connecting to the source filestore.

FTAM Parameters (cont.)		
parameter	**syntax**	**description**
ft_src_fname	string	the name of the source file to be transfered.
ft_dst_fname	string	the name of the destingation file for the transfer. Also name used on non-transfer calls.
ft_alt_src_fname	string	this file name is exchanged with ft_src_fname1 when the special "invert" test case is done.
ft_alt_dst_fname	string	this file name is exchanged with ft_dst_fname1 when the special "invert" test case is done.

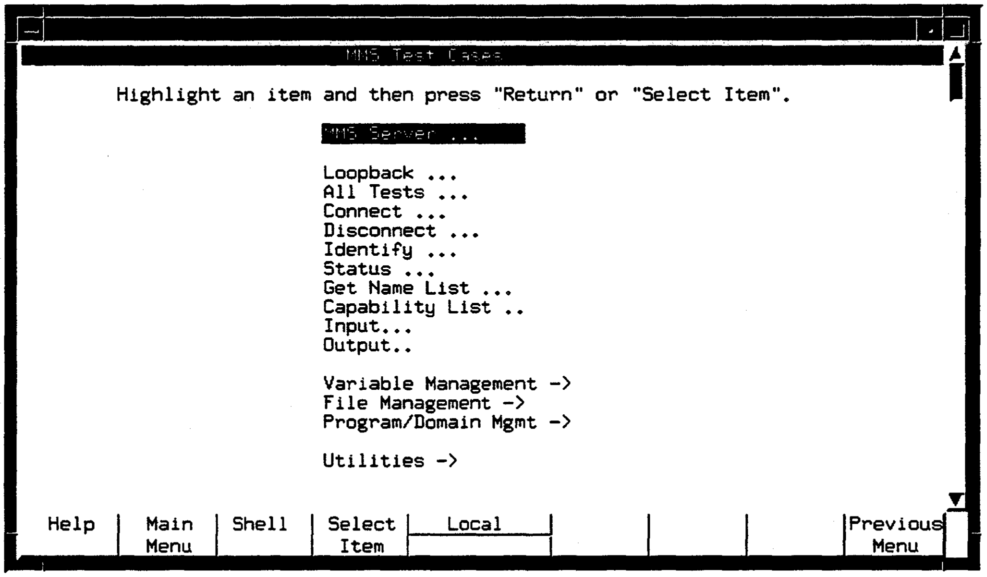
MMS Tests

Osidiag allows you to verify the ability to create and receive MMS connections with a remote. You can also utilize some of the VMD management, variable management, file management, program management and domain management facilities provided by MMS.

NOTE: many of the operations provided through osidiag are only supported as initiator. This means, for instance, that the osidiag Server test will not respond to a domain download request from a remote. The operations which have this restriction are noted with **Initiator Only** in the test case descriptions below. Also the operations the Server allows are listed under the description of that test.

Test Menu

The test case menu displayed for the MMS tests is shown below:



```
                              MMS Test Cases

      Highlight an item and then press "Return" or "Select Item".

                          MMS Server ...

                          Loopback ...
                          All Tests ...
                          Connect ...
                          Disconnect ...
                          Identify ...
                          Status ...
                          Get Name List ...
                          Capability List ..
                          Input...
                          Output..

                          Variable Management ->
                          File Management ->
                          Program/Domain Mgmt ->

                          Utilities ->

  Help  | Main | Shell | Select | Local |  |  |  |Previous
        | Menu |       | Item   |       |  |  |  | Menu
```

Server

This operation causes osidiag to listen for an inbound MMS connection indication from the remote node. Osidiag will respond positively to the connection, if received. Osidiag will then continue to listen for and respond to other MMS indications received until the connection is released or an error is detected.

If you wish to receive only a single MMS primitive and then be returned to the MMS menu, go to the Utilities menu and select "Save Connections" before initiating MMS activity.

Osidiag will listen for a default of 30 seconds, before giving up on receiving a connection (or other indication). This value can be changed under the Utilities menu or via the "-w" flag on the command line.

Osidiag will prompt for definitions of a local MMS variables when this test is selected. These may be left blank unless you will be servicing variable access requests. See the description below of read and write variable servicing.

The osidiag server supports the following operations. Requests outside this list will be rejected.

- Connect - a positive response is sent on receipt.
- Identify - the identify response is sent automatically by the MMS Service Provider Process (SPP), without intervention by osidiag.
- File Management - the MMS operations for file management are handled automatically by the MMS SPP, without intervention by osidiag.
- Read Variable - read variable requests are serviced by osidiag. The name of the variable is compared against the names of the three variables osidiag allows the user to configure. If the name matches any of these, then a positive response is sent, otherwise an error is given.
- Write Variable - write variable requests are serviced by osidiag. The name is checked as in variable reads. The type of the data being written must match that of the defined variable, or an error is returned.
- Conclude - the request to release the connection is confirmed by osidiag.

Loopback

This operation will cause osidiag to establish an MMS connection with itself and then release the connection.

Successful completion of this test indicates that the local node is configured correctly and capable of providing the MMS service.

All Tests

This operation will result in osidiag attempting to connect to the specified remote and then executing all of the MMS Client tests which it indicates support for. The supported functions are determined via the "Services Supported" parameter returned on connection.

The order of operations (if all supported) is as follows: **Connect, Identify, Status, Obtain File, Read File, Delete File, Write Variable, Read Variable, Download, Upload, Create Program, Start Program,** and **Stop Program,**

Connect

The connect test will result in a connection request being sent to the remote address you specify. By default the connection will be released immediately after it is successfully created.

If you do not want osidiag to close the connection after opening it, go to the Utilities menu first and select "Save Connections". This will result in the Connection test opening the connection and then returning you to the menu. You can then perform other MMS operations over the same connection.

NOTE: MMS provides support for two versions. By default the International Standard (IS) version is proposed on the connection (version 1). If the remote only supports the Draft International Standard (DIS), version 0, then you should go to the Utilities menu and modify the prompt level, so that osidiag allows you to override version defaults.

Disconnect

This operation is only meaningful if you are in "Save Connections" mode, which is controlled under the Utilities menu. It will disconnect a connection which was opened on a previous test.

Identify

This operation will cause osidiag to issue an Identify request to the remote.

Upon successful completion the Vendor Name, Model Name and Software Revision Number will be displayed.

Status

This operation will cause osidiag to request the status of the remote device.

Upon successful completion the Logical Status and Physical Status of the remote will be displayed.

Osidiag supports this as an **initiator only.**

Get Name List

This operation will issue a Get Name List request to the remote. The response will list the names of variables, domains, programs, types or operator stations defined at the remote node.

Osidiag supports this as an **initiator only.**

Osidiag will ask for the following parameters to perform this operation:

- Object Class
- Object Scope
- Domain Name

Capability List

This operation will issue a Capability List request to the remote. This response is a list of Capabilities which are visible strings whose meaning is defined by the remote.

Osidiag supports this as an **initiator only.**

Input

This operation will issue an Input request to the remote. This will result in the operator console on the remote prompting for input. When the input is entered the response will carry the entered data.

Osidiag supports this as an **initiator only.**

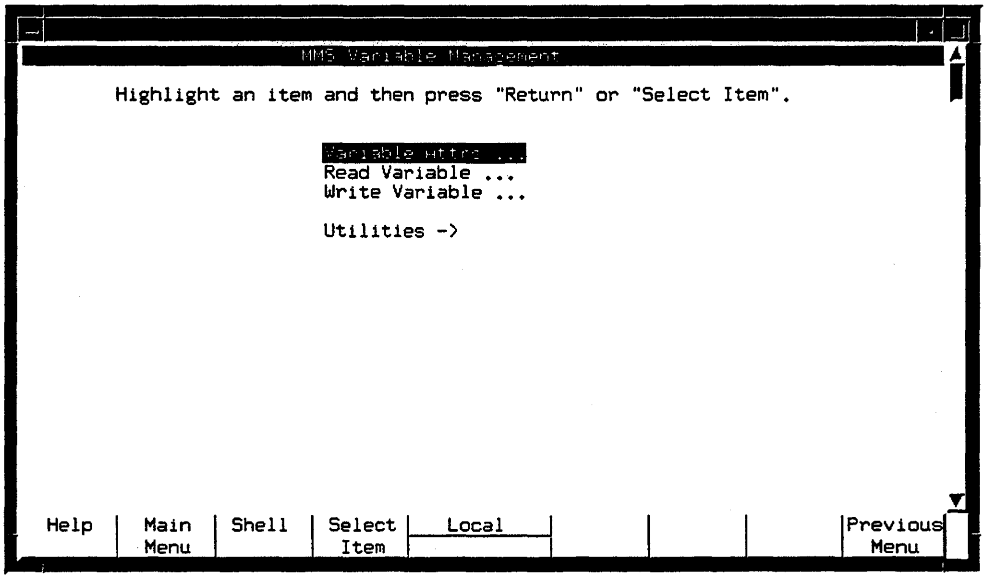
Output

This operation will issue an Output request to the remote. This will result in the operator console on the remote displaying the output specified.

Osidiag supports this as an **initiator only.**

Variable Management

This selection takes you to the following menu for performing MMS variable operations.



```
Highlight an item and then press "Return" or "Select Item".
```

Variable Attributes

This operation results in osidiag requesting the MMS attributes of the specified variable.

The following information is reported:

- Deleteable - Boolean (for remotely defined variables).
- Type - information about the variable type is displayed.

Read Variable

Selecting this menu item results in osidiag issuing a Read Variable request to the remote. You are prompted for the variable name and addressing type.

Upon successful completion the value of the variable will be displayed.

NOTE: only integer, visible string and octet string variables are supported. Other variable types will be partially displayed.

Write Variable

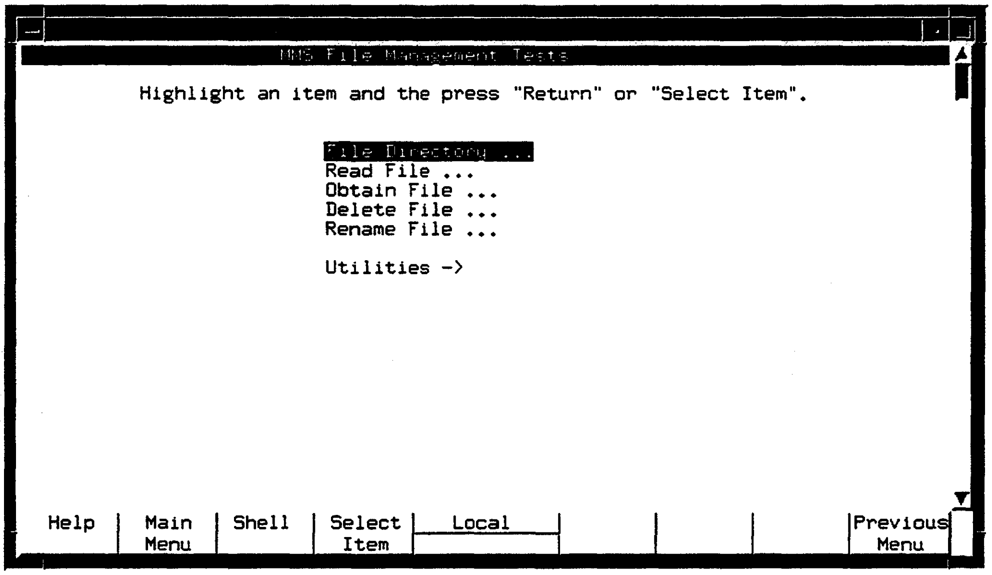
Selecting this menu item results in osidiag issuing a Write Variable request to the remote. You are prompted for the variable name, addressing type and value.

Upon completion the write response contents will be displayed.

Only integer, visible string and octet string variables are supported.

File Management

This will take you to the menu shown below to perform the various MMS file management operations. MMS file management uses a different protocol than FTAM and requires a remote MMS application capable of servicing these requests. The "Server" function under osidiag serves this role.



```
                         MMS File Management Tests

        Highlight an item and the press "Return" or "Select Item".

                         File Directory ...
                         Read File ...
                         Obtain File ...
                         Delete File ...
                         Rename File ...

                         Utilities ->
```

File Directory

Selecting this menu item results in osidiag issuing a File Directory request to the remote. If successful the contents of the directory will be reported.

NOTE: if the contents of the remote directory are large, the complete list may not be delivered in one response. Osidiag will indicate whether the list is complete by showing the line "more follows: TRUE/FALSE" at the end of the output. If the TRUE, the list is only partial.

Read File

Selecting this menu item results in osidiag attempting to read a file from the remote to the local file specified.

Obtain File

Selecting this menu item results in osidiag issuing an Obtain File request to the remote. The remote will then perform the Read File sequence to copy a file from the local node to the remote. The confirmation to the obtain file indicates the remote's success of copying the file.

Delete File

Selecting this menu item results in osidiag issuing an Delete File request to the remote. If successful the file specified is deleted from the remote filestore.

Rename File

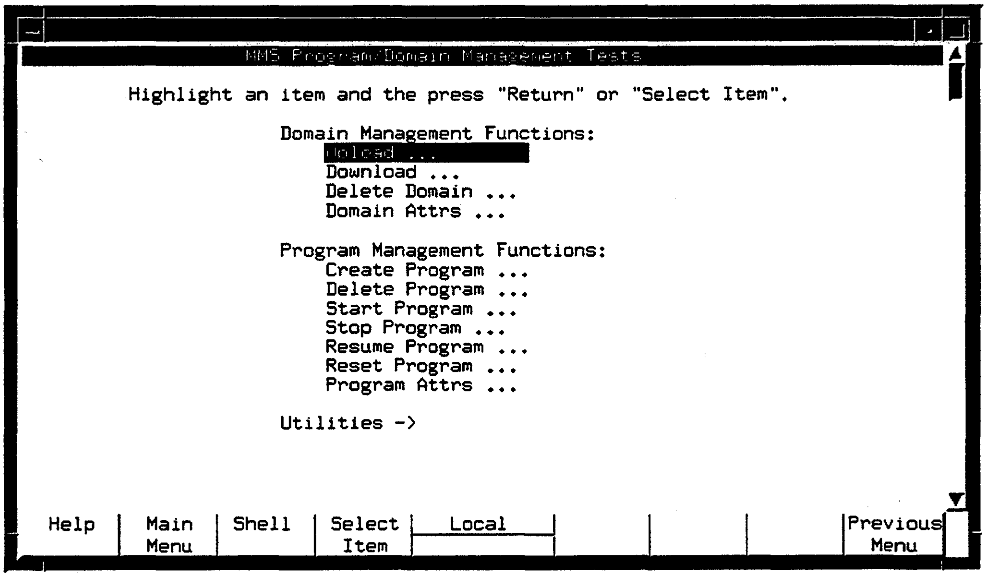
Selecting this menu item results in osidiag issuing a Rename File request to the remote. If successful the file specified is renamed on the remote filestore.

Program/Domain Management

This takes us to the menu shown below. Program management allows an MMS application to control the execution of programs on a remote system. Domain management allows an MMS application to upload and download domains (typically portions of programs) between the local and the remote.

Osidiag acts as an **initiator only** for all of these functions. HP's MMS library does however allow application developers to write responder code for these operations.

A common use for these facilities is directing the behavior of Programmable Logic Controllers (PLC's). These devices in turn control or monitor things such as valves, power switches, thermostats, flow meters, etc.



```
                    MMS Program/Domain Management Tests

         Highlight an item and the press "Return" or "Select Item".

                      Domain Management Functions:
                          Upload ...
                          Download ...
                          Delete Domain ...
                          Domain Attrs ...

                      Program Management Functions:
                          Create Program ...
                          Delete Program ...
                          Start Program ...
                          Stop Program ...
                          Resume Program ...
                          Reset Program ...
                          Program Attrs ...

                      Utilities ->
```

Upload

Selecting this menu item results in osidiag issuing an Upload request to the remote. The remote will transfer the content of the named domain to osidiag, which will place it in a local file which you specify.

Download

Selecting this menu item results in osidiag issuing an Download request to the remote. Osidiag will transfer the content of a local file you specify to be used as the domain on the remote.

Delete Domain

Selecting this menu item results in osidiag issuing a Delete Domain request to the remote. If successful the domain specified will be removed.

The domain should not be part of any active program invocations. I.e. you should delete any program invocation which contains this domain before deleting the domain itself.

Domain Attributes

Selecting this menu item results in osidiag requesting the MMS attributes of the specified domain.

The following information is reported:

- Capability List - ASCII information, meaning determined by remote.
- State - the state of the domain (i.e. loading, ready, in use).
- Deletable, Shareable - Boolean flags indicating if these attributes are true.
- Program List - program invocations this domain is part of (i.e. through Create Program).

Create Program

Selecting this menu item results in osidiag issuing an Create Program request to create a named program which can then be executed on the remote PLC.

A program is created from a set of domains which may have previously been downloaded to the PLC.

Delete Program

Selecting this menu item results in osidiag issuing a Delete Program request to the remote. If successful the program invocation will be removed.

The program should have been previously stopped for this operation to succeed.

Start Program

Selecting this menu item results in osidiag issuing a Start Program request to the remote. If successful the program named will be started.

A named program should initially be created through the "Create Program" menu selection.

Stop Program

Selecting this menu item results in osidiag issuing a Stop Program request to the remote. If successful the program named will be stopped.

The program should be currently running for this operation to succeed.

Resume Program

Selecting this menu item results in osidiag issuing a Resume Program request to the remote. If successful the program invocation will resume execution.

The program should have previously been stopped for this operation to succeed.

Reset Program

Selecting this menu item results in osidiag issuing a Reset Program request to the remote. If successful the program invocation will reset to the same state as after a Create Program request.

This operation is necessary after a Stop Program operation in order for the Start Program operation to succeed.

Program Attributes

Selecting this menu item results in osidiag requesting the MMS attributes of the specified program.

The following information is reported:

- Program State - (i.e. running, idle, stopped, etc).
- Domain List - the domains which make up this program.
- Deletable, Reusable, Monitor - Boolean flags.

Utilitites

This will take you to the Utilities menu.

Parameters

The following describe the parameters which the various MMS tests require.

MMS Server Presentation Address

A data entry window is presented by osidiag in order to get the address of the remote. It looks the same as the FTAM Presentation Address screen. See the "FTAM" section for an example.

The default value presented by osidiag is that configured for the first local MMS application found in the configuration file "local_app" whose AE Title matches osidiag's criteria. The criteria used by osidiag is that the name contain an attribute whose value is either "MMS" or "mms". If osidiag finds no such configured values a warning is given and no default is used.

MMS Server DDN

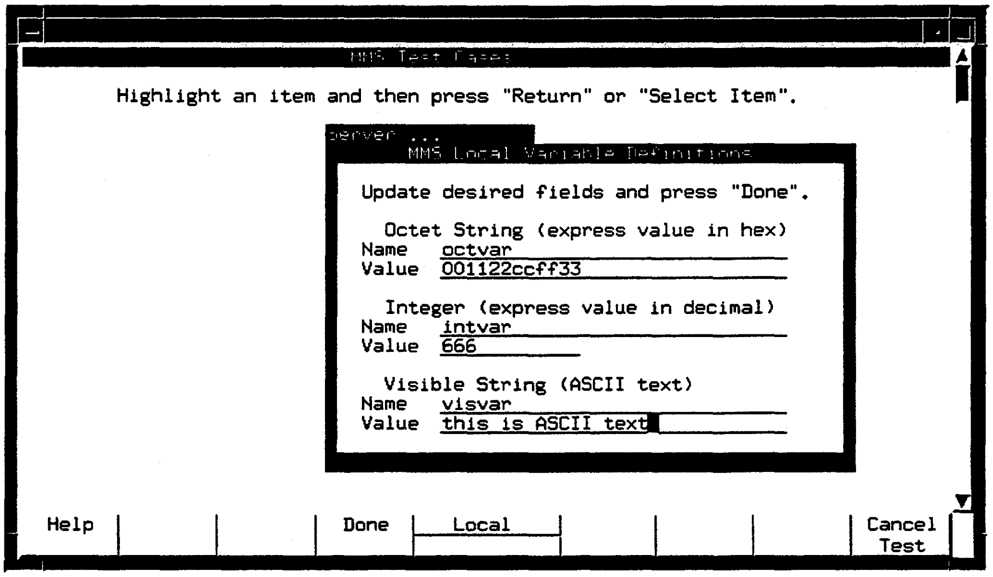
By default osidiag will ask you for a presentation address in order to reference the remote. If you wish to specify the Directory Distinguished Name instead you may. To do this you must modify the "Prompt Level" under the Utilities menu to enable prompting for remote DDN's.

The window presented looks like that used for FTAM. See the "FTAM section for an example.

MMS Local Variable Definitions

The MMS local variables are prompted for each time time you run the Server test. The definitions you provide are used when an incoming read variable or write variable indication is received by the Server test.

The definition screen appears as follows:



```
                         MMS Test Cases

     Highlight an item and then press "Return" or "Select Item".

                           Server ...
                              MMS Local Variable Definitions

                           Update desired fields and press "Done".

                             Octet String (express value in hex)
                           Name   octvar
                           Value  001122ccff33

                             Integer (express value in decimal)
                           Name   intvar
                           Value  666

                             Visible String (ASCII text)
                           Name   visvar
                           Value  this is ASCII text
```

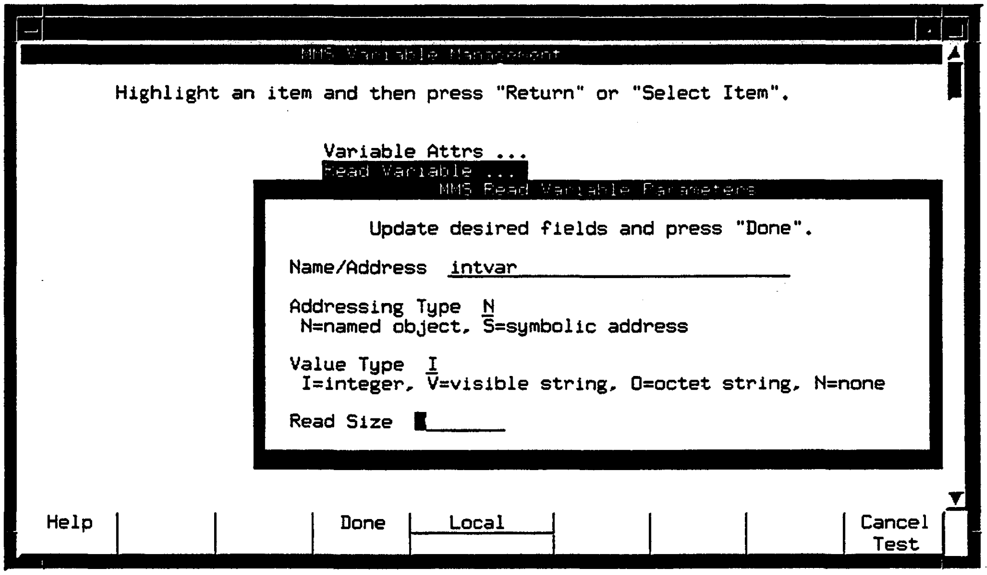
To "define" a variable, you must give it a name. Then this name will can be matched against the name specified on an inbound read or write request.

You can define a variable of each of the following types:

- Octet String - this is a list of octets (bytes) whose value you specify. The value is expressed in hex. Alternatively you may use a quoted ASCII string.
- Integer - this is a 32 bit signed integer value.
- Visible String - this is ASCII text.

Read Variable Parameters

When issuing a variable read request, the following screen will be displayed.



```
                         MMS Variable Management
     Highlight an item and then press "Return" or "Select Item".

                           Variable Attrs ...
                           Read Variable ...
                             MMS Read Variable Parameters

                     Update desired fields and press "Done".

                 Name/Address  intvar

                 Addressing Type  N
                  N=named object, S=symbolic address

                 Value Type  I
                  I=integer, V=visible string, O=octet string, N=none

                 Read Size
```

The fields have the following meaning:

- Name/Address - this should be the name of the variable to be read on the remote node.
- Addressing Type - indicates whether the variable is to be accessed as a named object, or whether a symbolic address will be passed. Named objects should be used unless the remote only supports symbolic addresses.
- Value Type - indicates what the type of the variable to be read is. If you use a named object, the value type is ignored. If symbolic addressing is used then you should either specify none or the corresponding type for the remote.
- Read Size - indicates the amount of data to be read. This is only used with symbolic addressing.

Example

The following shows a successful MMS loopback test.

```
entity:    MMS
test case: Loopback
------------------------------------
mm_my_ddn           : /O=hpindon/AP=mms/AE=client1
mm_version          : 1
mm_remote_ddn       : /O=hpindon/AP=mms/AE=client1
------------------------------------
mm_vactivation()       vmd_id:   0x00000004   result: 0
mm_aeactivation()      ae_label: 0x186d0009   result: 0
mm_listen()            ae_label: 0x186d0009   result: 0
mm_connect()           ae_label: 0x186d0009   result: 0
em_wait()              ae_label: 0x186d0009   result: 0
  COMPLETED:  mm_listen()             conn_id:  0x00000101   result: 0
mm_answer()            conn_id:  0x00000101   result: 0
em_wait()              ae_label: 0x186d0009   result: 0
  COMPLETED:  mm_connect()            conn_id:  00000000   result: 0
mm_cnclude()           conn_id:  00000000   result: 0
mm_ireceive()          conn_id:  0x00000101   result: 0
em_wait()              ae_label: 0x186d0009   result: 0
  COMPLETED:  mm_ireceive()           ae_label: 0x186d0009   result: 0
                       ind type: MM_CONCLUDE_IND
mm_coresponse()        conn_id:  0x00000101   result: 0
em_wait()              ae_label: 0x186d0009   result: 0
  COMPLETED:  mm_cnclude()            ae_label: 0x186d0009   result: 0
mm_slisten()           ae_label: 0x186d0009   result: 0
mm_aedeactivation()    ae_label: 0x186d0009   result: 0
mm_vdeactivation()     vmd_id:   0x00000004   result: 0
------------------------------------
version:                1
max segment size:       4096
nesting level:          4
max services calling:   1
max services called:    1
variable parameters supported:
                        MM_STR1
                        MM_STR2
                        MM_VNAM
                        MM_VADR
services supported:
                        MM_IDENTIFY
                        MM_READ
                        MM_WRITE
                        MM_DOWNLOAD_SEGMENT
                        MM_TERMINATE_DOWNLOAD_SEQUENCE
```

```
                                MM_OBTAIN_FILE
                                MM_FILE_OPEN
                                MM_FILE_READ
                                MM_FILE_CLOSE
                                MM_FILE_RENAM
                                MM_FILE_DELETE
                                MM_FILE_DIRECTORY
                                MM_CONCLUDE
-----------------------------------
test status:  PASSED
========================================================================
```

The first two lines indicate the test being executed. The next section of output is the parameter display. The third section is the dialog display. The fourth section shows the Negotiated Parameters. Lastly the test status is shown.

Parameter Display

The parameters used for the connection test are displayed after the test case name. The meaning of the parameters is described in the table at the end of this section.

Dialog Display

The following MMS primitives are executed to perform a connection test. More information about the various MMS primitives may be found in the MMS Programmer's Guide or in the online manual pages.

- **mm_vactivation()** - initializes VMD tables local to the osidiag process. No network activity is generated.
- **mm_aeactivation()** - this creates a "Server Provider Process" (see "Architecture" diagram at beginning of "Concepts" section). The SPP is responsible for the actual encoding and decoding of MMS PDU's it also interacts with the stack. This operation does not result in any network activity.
- **mm_listen()** - posts a listen for an inbound connection indication. This call is made asynchronously for the loopback test. Otherwise we would pause until a connection indication was received.
- **mm_connect()** - issue a connection request, which is directed at ourselves (the local ddn and remote ddn are the same). This call is also made asynchronously for the loopback test. Otherwise the connect call blocks until a confirmation is received or an error is encountered.
- **em_wait()** - this waits for an asynchronous operation to complete. In the successful case, our mm_listen() call completes after receiving the connect indication caused by calling mm_connect().

- **mm_answer()** - respond positively to the connection indication received.
- **em_wait()** - get the connect confirmation which completes the call to mm_connect().
- **mm_cnclude()** - issue a conclude request to terminate the connection. This is done asynchronously for loopback. Otherwise this call blocks until the conclude confirmation is received.
- **mm_ireceive()** - listen for an indication on an established connection. Indications may be variable access, abort, or conclude operations. Recall that osidiag only supports a limited number of services as a responder. This call is made asynchronously, so that we can control the maximum amount of time we will wait for a response.

  The next three lines show the ASNYCH mm_ireceive() completing. The type of indication received is displayed. In this case it is a conclude indication.
- **mm_coresponse()** - respond to the conclude indication we received.
- **em_wait()** - await the completion of our mm_cnclude() request.
- **mm_aedeactivation()** - this call terminates the SPP. It does not result in any network activity. Errors on earlier operations may cause this to fail.
- **mm_vdeactivation()** - release any tables set up in the local process for the VMD. No network activity.

Negotiated Parameters

At the completion of an MMS test, the parmaters negotiated with the remote responder are displayed.

- **version** - this is the version of MMS which will be used. A value of 1 indicates IS, 0 indicates DIS.
- **max segment size** - this is the maximum size of an MMS PDU. Osidiag will propose the default value configured via osiconf.
- **nesting level** - this is the maximum nesting depth of any structured variable. A value of 0 indicates that structured variables are not allowed. Osidiag will propose the default value configured via osiconf. Note for osidiag's purposes only simple variables are used, so this is OK.
- **max services calling** - this is the maximum number of service requests that may be queued without awaiting confirmation to the entity which requested the connection. Note that osidiag always waits for each service to be confirmed.
- **max services called** - this is the maximum number of service requests that may be queued without awaiting confirmation to the entity which accepted the connection.
- **variable parameters supported** - this indicates what types of variables have been negotiated. Those which are supported are displayed. Possible values are MM_STR1, which indicates support for structured type 1 variables (arrays); MM_STR2, structured type 2 variables (arbitrary structures); MM_VNAM, named variables, MM_VADR, addressed variables, MM_VLIS, lists of variables.
- **services supported** - this indicates what MMS services are supported by the remote

entity, note that the list given by HP corresponds to the functions described as being supported by the server. One exception is the inclusion of MM_DOWNLOAD_SEGMENT and MM_TERMINATE_DOWNLOAD_SEGMENT. This does not indicate that osidiag can receive downloads, rather it is there because during the download dialog it is the entity being downloaded which initiates these two requests.

MMS Variable Information

The following shows a successful variable write test.

```
entity:    MMS
test case: Write Variable
-------------------------------------
mm_my_ddn            : /O=hpindon/AP=mms/AE=client1
mm_remote_paddr      : 0002.0002.0002."hpindon"
mm_var_addr          : intvar
mm_var_data          : 50
mm_addr_type         : N
mm_val_type          : I
-------------------------------------
mm_vactivation()      vmd_id:   0x00000005   result: 0
mm_aeactivation()     ae_label: 0x1ddc0009   result: 0
mm_connect()          conn_id:  00000000   result: 0
mm_write()            conn_id:  00000000   result: 0
mm_cnclude()          conn_id:  00000000   result: 0
mm_aedeactivation()   ae_label: 0x1ddc0009   result: 0
mm_vdeactivation()    vmd_id:   0x00000005   result: 0
-------------------------------------
name of variable:      intvar
data value:            50
data kind:             INTEGER
data size:             32
nature of variable:    NAMED OBJECT
object scope:          VMD SPECIFIC
access error:          SUCCESS
-------------------------------------
test status:  PASSED
========================================================================
```

Osidiag displays the following information about a variable in the following situations:

- mm_write(), mm_read() completed - when a write or read request is completed, the information sent out and received are displayed.
- write or read indication received - when an indication is received in server mode, the content of that indication is shown.

- write or read response sent - when the server responds to the indication (either successfully or with error) the information sent back is shown.

The following components may be shown:

- **name of variable** - the name of the variable specified in the operation.
- **data value** - the value of the variable being accessed. This value will not be shown when a read indication is received, or the read response fails.
- **data kind** - indicates the type of variable being accessed. Osidiag supports INTEGER, OCTET_STRING and VISIBLE_STRING types only.
- **data size** - gives the size of the data value. For an integer variable, this is the number of bits of precision. For the other two types, this is the length in bytes.
- **nature of variable** - this corresponds to the mm_addr_type parameter you set. It indicates if we have a named variable, an address, or a address and type variable.
- **object scope** - indicates the scope of the named variable. Osidiag will generate a VMD specific named object for its requests. Other values are Association Specific, which means the variable is only accessible over this connection; and Domain Specific which is not supported by HP as a responder.
- **address kind** - indicates what type of address we are using. Osidiag will always generate SYMBOLIC addresses. The other kinds are NUMERIC and UNCONSTRAINED.
- **type kind, type class, type size** - these will be shown for ADDRESS AND TYPE variable specifications only. It means that our read request explicitly carries the type of variable we are trying to access. In the case of NAMED OBJECT and ADDRESS access, the type is implied by the address given.
- **access error** - this gives any error associated with the variable access. There are three values for this which are acceptable: **SUCCESS, OBJECT ACCESS DENIED** and **TRANSFORM ERROR.** The first indicates that we successfully performed our operation. The second may reported by the server when an indication is received, the last may be reported when a read operation completes. These two errors are due to the fact that osidiag does not define variables to the MMS interface. This allows osidiag to more completely display variable information.

Interpreting Errors

If an MMS test operation fails, the following output will appear in the status section at the end of the output:

```
------------------------------------
test status:  FAILED
operation:    mm_connect()
return code:  4325625
vendor code:  0
error code:       MMC199_PRES_SERVICE_PROVIDER

(MME249) Rejected permanent
No additional information available

==========================================================================
```

The meaning of each field is as follows:

- **test status** - indicates whether we passed or failed. Any error encountered during the operation (including setting up tracing and logging, if requested) will result in a status of failed.
- **operation** - indicates at what point in the test we encountered the first error. Typically this will be the name of an MMS operation.
- **return code** - this is the return code from the first error encountered. For MMS errors this is always a large number because MMS errors start with the base error number of 66<<16, which is 4325376. The meaning of this error code is displayed below the "additional desc:" field. In this case it is "(MME249) Rejected permanent".
- **vendor code** - this field may contain additional information about the error. Any text associated with it will be displayed immediately after the return code text. In this case it is "No additional information available".
- **mask** - this field will only be present if we have gotten more error information returned from the call. It is a bit mask with the three low order bits are significant "-----DAE". The D bit indicates that there is an "additional description", the A bit indicates that there is an "additional error code" and the E bit indicates that there is an "error code" in addition to the return and vendor codes.
- **error code, additional error, additional desc** - these may contain more information describing the error. In this case we are told that the Presentation Service Provider is the entity which rejected the connection. These values will be conditionally displayed according to the mask value.
- **Formatted error text** - after the above fields are displayed, osidiag will provide any textual error information which is available. If there is a "log instance" given, then this may be matched to an error in the nettl log file.

Further Information

Further information on interpreting errors may be found in the Troubleshooting section of this manual. Also consult the "MMS Reference Manual."

Parameter Summary

The following table summarizes the MMS parameters which are configurable by the caller of osidiag.

MMS Parameters		
parameter	**syntax**	**description**
mm_remote_paddr	paddr	specifies the presentation address of the remote MMS Server. Will override the mm_remote_ddn if both are given.
mm_my_ddn	ddn	specifies the directory distinguished name associated with the local MMS Ae Invocation. Used for Client, Server and loopback mode.
mm_remote_ddn	ddn	specifies the ddn of the MMS Server which services requests from the Client.
mm_domain_name	string	specifies the domain name on the remote PLC to be used for uploading and downloading.
mm_local_rd_fname	string	specifies the name of a local file used for the peer to peer read file operation.
mm_remote_rd_fname	string	specifies the name of the remote file used for peer to peer transfers.
mm_local_ob_fname	string	specifies the name of a local file to be read by the remote on the peer to peer obtain file call.
mm_remote_ob_fname	string	specifies the name of the remote file which will be created on the peer to peer obtain file call. Also used for rename.
mm_delete_fname	string	the name of the remote file to be deleted.
mm_upload_fname	string	specifies the local file to store an upload to.
mm_download_fname	string	specifies the local file to perform a download from.
mm_prog_id	string	specifies the program invocation identifier to be used on the program invocation management commands.

MMS Parameters (cont.)		
parameter	**syntax**	**description**
mm_var_addr	string	specifies the name (or address) of the remote variable to be accessed.
mm_addr_type	atype	specifies the type of addressing to be performed. (S or N)
mm_val_type	vtype	specifies the type of the data value being assigned or read from the variable. (I, V, O or N).
mm_var_data	var_data	specifies the value of the variable to be used on the write. The specification must be consistent with the mm_val_type chosen.
mm_var_size	var_data	specifies the size of the variable being accessed. Only required if mm_addr_type = S and mm_val_type != N.
mm_oct_name	string	specifies the name of an octet string variable which the MMS Server will allow reads and writes to be performed against.
mm_oct_val	var_data	specifies the initial value of the octet string variable which the MMS Server manages. If the remote client performs a write this value will be altered.
mm_int_name	string	specifies the name of an integer variable which the MMS Server will allow reads and writes to be performed against.
mm_int_val	var_data	specifies the initial value of the integer variable which the MMS Server manages. If the remote client performs a write this value will be altered.
mm_vis_name	string	specifies the name of a visible string variable which the MMS Server will allow reads and writes to be performed against.
mm_vis_val	var_data	specifies the initial value of the visible string variable which the MMS Server manages. If the remote client performs a write this value will be altered.

MMS Parameters (cont.)		
parameter	**syntax**	**description**
mm_obj_scope	otype	specifies the scope of the objects to be referenced. Typically the scope is VMD specific. (V, D or A)
mm_rename_fname1	string	the old name of a file to be renamed.
mm_rename_fname2	string	the new name of the file to be renamed.
mm_rename_fname2	string	the new name of the file to be renamed.
mm_station	string	the name of the operator station which will receive the input or output operation.
mm_inout_data	string	the text which accompanies an MMS input or output operation.
mm_dir_name	string	the name of the directory to be listed by the MMS directory operation.
mm_obj_class	class	the type of objects to be listed by a MMS get name list operation. (VAR, TYPE, PROG, DOM, OPER).
mm_version	integer	the MMS version to be used for this test (DIS=0, IS=1).

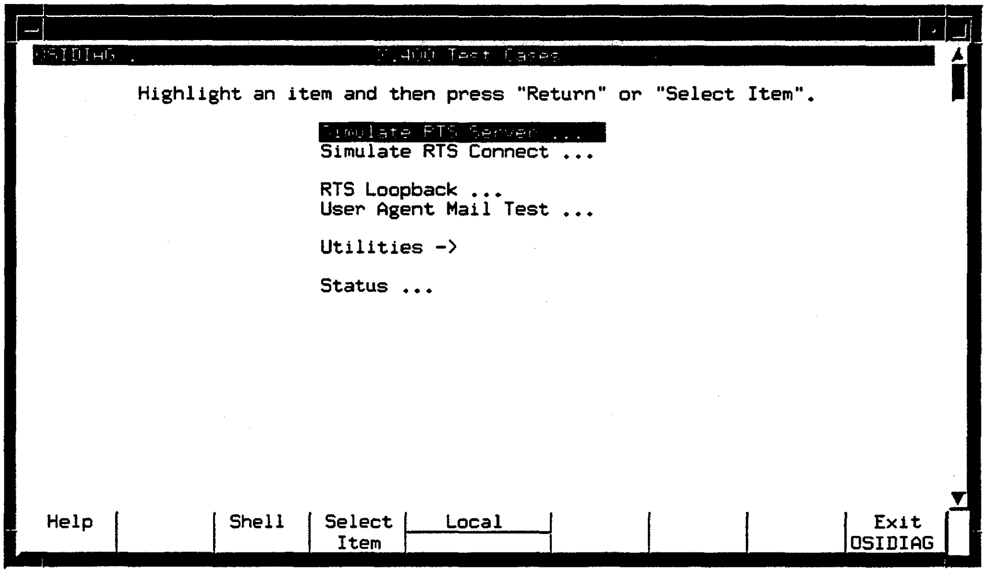
X.400 Tests

Osidiag allows you to test X.400 at two levels. First at the mailx interface level. And second at the Reliable Transfer Service (RTS) level. In the case of RTS there is a loopback test which invokes the actual X.400 RTS and there are connect and server tests which simulate the RTS.

X.400 tests are also available under the Diagnostics menu of x4admin(1M). Other tests are available from the command line. Consult the "Managing HP X.400" manual for more information about these tests.

Test Menu

The test case menu displayed for the X.400 tests is shown below:



```
OSIDIAG .                    X.400 Test Cases

        Highlight an item and then press "Return" or "Select Item".

                             Simulate RTS Server ...
                             Simulate RTS Connect ...

                             RTS Loopback ...
                             User Agent Mail Test ...

                             Utilities ->

                             Status ...

  Help  |        | Shell | Select |  Local  |       |       |       | Exit
        |        |       |  Item  |         |       |       |       |OSIDIAG
```

Simulate RTS Server

The server test will create a pseudo RTS entity which will listen for an inbound connection from a remote RTS.

X.400 must be down on the local node when running this test.

This RTS will listen at the same addreess as the real X.400 RTS. It will wait for a default duration of 30 seconds. If after 30 seconds no inbound connection is received the test will terminate with a timeout failure. The timeout duration can be modified by going to the utilities menu and selecting "Wait Time" prior to the execution of the test.

Simulate RTS Connect

The connect test will result in a connection request being sent to a remote RTS. After a confirmation is received from the remote the connection is dissolved.

RTS Loopback

This test will cause the X.400 RTS to deliver a message to itself. This will test the ability of X.400 RTS to communicate with the stack and use the underlying network.

When running this test you will be prompted for the following parameters

- Link Type
- Local Address

User Agent Mail Test

This test will send a mail message to the specified recipient. The progress of the mail message will be traced in the various local event queues to keep you updated on the status of the message. The message will also be sent with receipt notification on, so that a response from the remote is anticipated and reported.

Utilities

By selecting this item you will be taken to the Utilities menu. To return from the Utilities to the X.400 tests press Exit.

Status

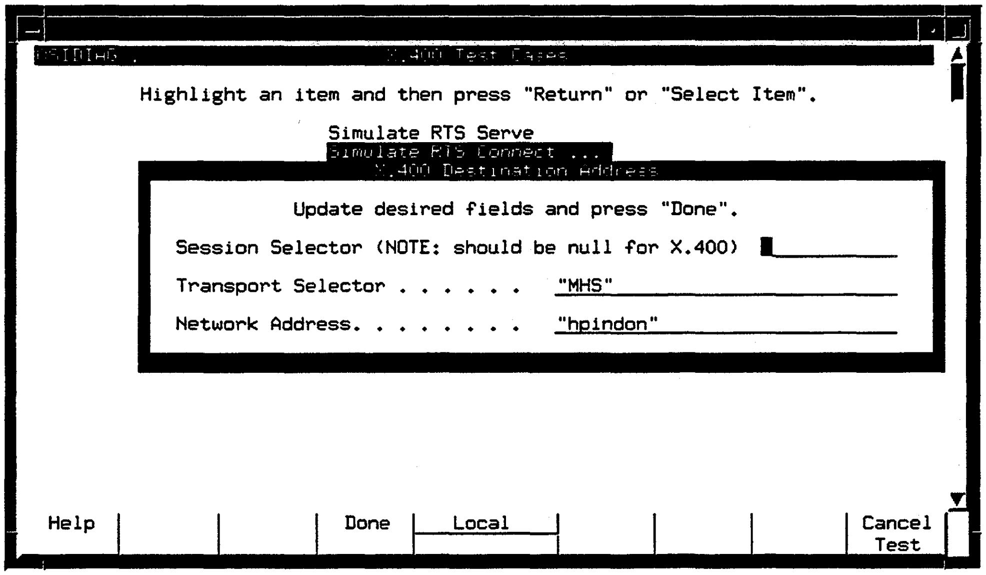
This option will invoke the program x4stat to report the status of the X.400 system and the underlying stack components. Various message queue statistics are also reported.

Parameters

The following describe the parameters which you will be prompted for when running the various X.400 tests.

Destination Address

The data entry window presented by osidiag in order to get the address of the remote is shown below:



```
OSIDIAG .                    X.400 Test Cases

        Highlight an item and then press "Return" or "Select Item".

                         Simulate RTS Serve
                         Simulate RTS Connect ...
                             X.400 Destination Address

                     Update desired fields and press "Done".

          Session Selector (NOTE: should be null for X.400) ________

          Transport Selector . . . . . .   "MHS"

          Network Address. . . . . . . .   "hpindon"
```

The Session address of the remote is composed of three components: the **session selector** the **transport selector** and the **network address**. These fields should contain the Session address which is configured for the remote RTS. For X.400 '84 the session selector value should be NULL.

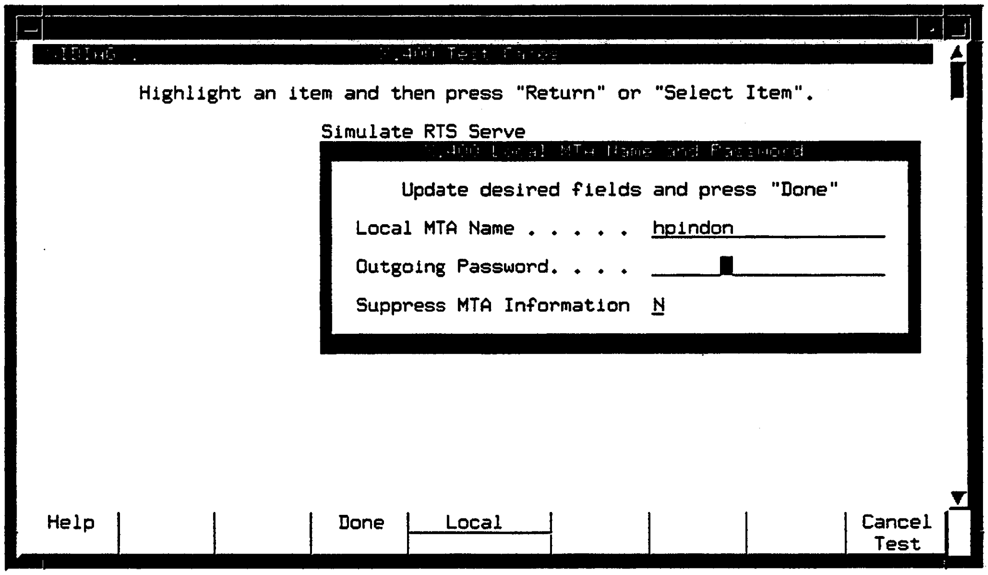
The default value presented by osidiag is ."**MHS.<local**NetAddr**>**" When running the Connect against the local RTS this will be sufficient.

When running against a remote HP node you need only change the Network Address. For non-HP nodes this value should be obtained by asking the network administrator for the remote node. This value should also be configured in the "Adjacent MTA's" section under x4admin.

Note that if the selector values are all ASCII that you may enter them as quoted strings. Otherwise specify them in hex.

Local MTA Name and Password

The following form is displayed after you have entered the remote addressing information requested.



Highlight an item and then press "Return" or "Select Item".

Simulate RTS Serve

Update desired fields and press "Done"

```
Local MTA Name . . . . .  hpindon
Outgoing Password. . . .
Suppress MTA Information  N
```

Local MTA Name

This is the MTA name assigned to our local node. Remote nodes validate incoming connections based on this name and the password provided.

Outgoing Password

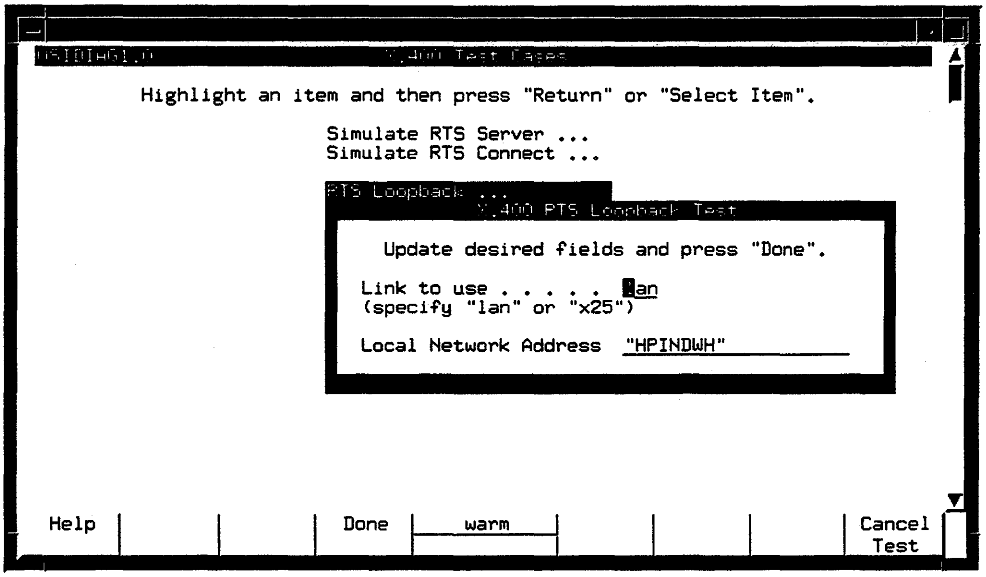
The password is checked by the remote in order to verify an RTS is who it indicates it is. The value should match the outgoing password password configured for this remote system.

Suppress MTA Information

The MTA Name and Password are optional fields in the connection request. Simply leaving the fields blank however results in zero length values being sent to the remote. These could potentially be considered meaningful by the remote. In order to send no MTA information set this field to "Y".

RTS Link Type

The following pop up form is presented on the loopback test to determine which link, LAN or X.25 you wish to loopback over.

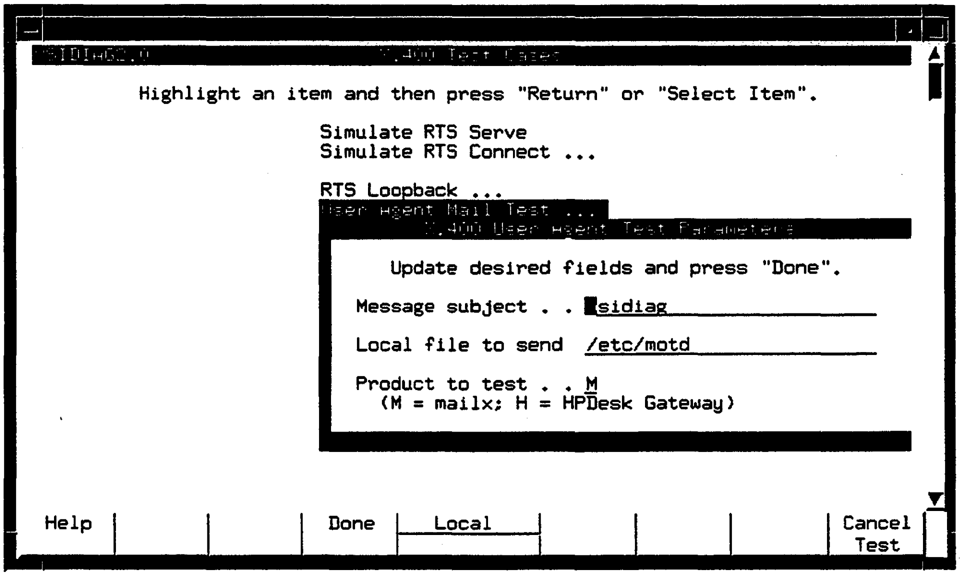


```
OSIDIAG1.0                    X.400 Test Cases

        Highlight an item and then press "Return" or "Select Item".

                       Simulate RTS Server ...
                       Simulate RTS Connect ...

                       RTS Loopback ...
                                X.400 RTS Loopback Test

                          Update desired fields and press "Done".

                         Link to use . . . . . . lan
                         (specify "lan" or "x25")

                         Local Network Address  "HPINDWH"
```

You are also prompted for the local network address associated with this link. Osidiag will fill in the default value. If you have both LAN and X.25 configured, then to test over X.25 you will need to determine the modify the Network Address. You can view the Addresses configured for the system by using the Status... test under the Transport or Session Test Cases.

X.400 User Agent Parameters

This window asks you for other parameters needed to perform the test. The defaults will suffice unless you have a particular text file you want delivered.



```
OSIDIAG2.0                    X.400 Test Cases

        Highlight an item and then press "Return" or "Select Item".

                         Simulate RTS Serve
                         Simulate RTS Connect ...

                         RTS Loopback ...
                         User Agent Mail Test ...
                                X.400 User Agent Test Parameters

                            Update desired fields and press "Done".

                          Message subject . . osidiag
                          Local file to send  /etc/motd
                          Product to test . . M
                            (M = mailx; H = HPDesk Gateway)
```

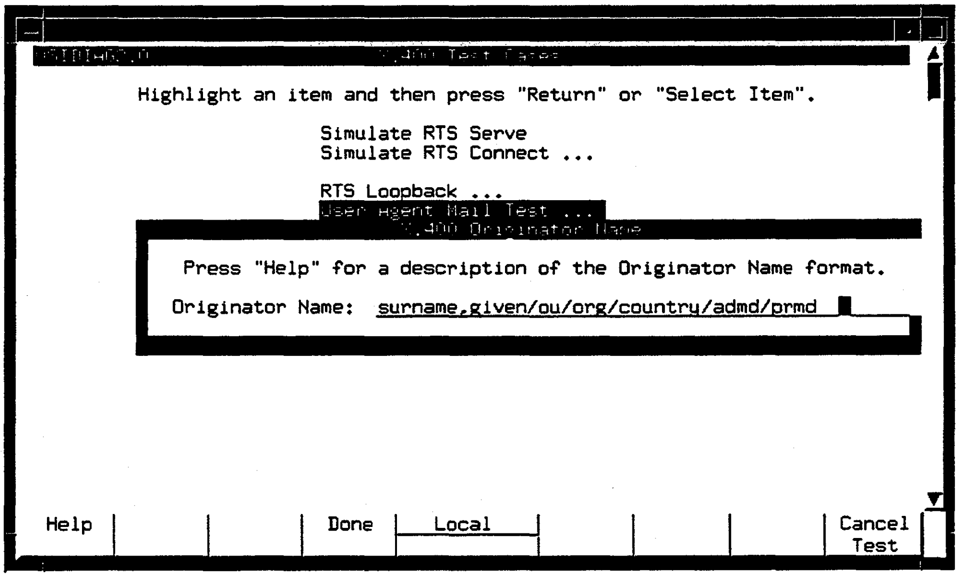
The **message subject** will be received on the remote as the subject of the message sent.

The **local file** identified will have its contents sent to the remote. The file "/etc/motd" is the HP-UX system message of the day and is typically present.

The **product to test** field identifies whether you wish to run the test against standard X.400 product through mailx, or if you are testing the HPDesk gateway.

X.400 Originator Name

This window is presented only if you are testing the HPDesk gateway product.



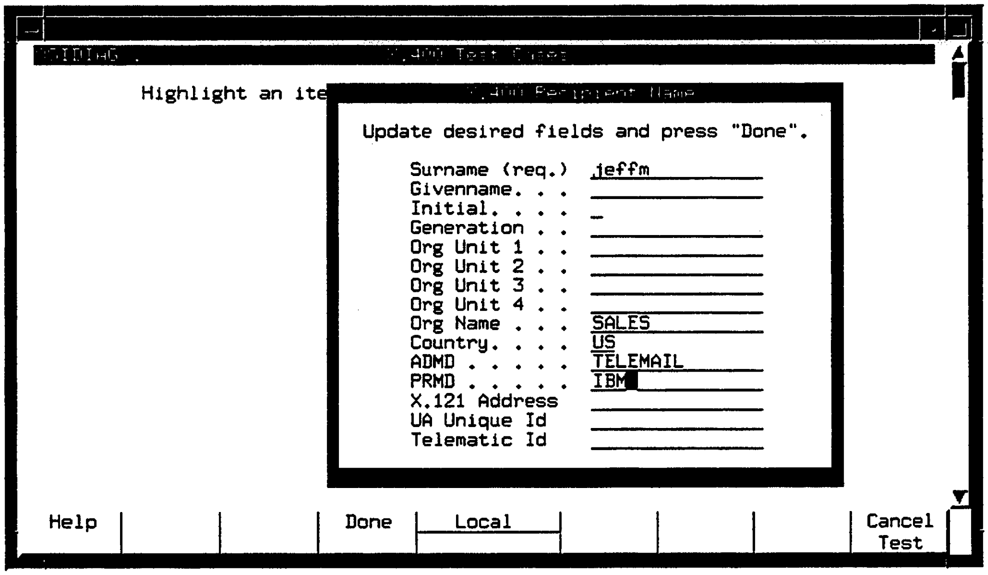
```
OSIDIAG.0                    X.400 Test Cases

        Highlight an item and then press "Return" or "Select Item".

                        Simulate RTS Serve
                        Simulate RTS Connect ...

                        RTS Loopback ...
                        User Agent Mail Test ...
                            X.400 Originator Name

          Press "Help" for a description of the Originator Name format.

         Originator Name:  surname,given/ou/org/country/admd/prmd
```

Originator information is required, because as a gateway, the originator information cannot be derived from local information. The syntax is shown in the example above.

The **message subject** will be received on the remote as the subject of the message sent.

X.400 Recipient Name

The following window is presented when you select the User Agent Mail Test it asks for the name of the recipient.



```
OSIDIAG .                    X.400 Test Cases

      Highlight an ite       X.400 Recipient Name

             Update desired fields and press "Done".

                 Surname (req.)  jeffm
                 Givenname. . .
                 Initial. . . .
                 Generation . .
                 Org Unit 1 . .
                 Org Unit 2 . .
                 Org Unit 3 . .
                 Org Unit 4 . .
                 Org Name . . .  SALES
                 Country. . . .  US
                 ADMD . . . . .  TELEMAIL
                 PRMD . . . . .  IBM
                 X.121 Address
                 UA Unique Id
                 Telematic Id
```

Recall the discussion of recipient names in the "Concepts" section. Fields which are typically used are the following.

- **Surname** - for HP-UX systems this is the login name of the individual.
- **Org Name** - a meaningful orginization name (i.e. SALES).
- **Country** - the ISO two letter country code for the recipient.
- **ADMD** - identifies the adminstrator of the X.400 network.
- **PRMD** - identifies any private sub administrator of addresses.

Examples

Successful Connection Test

The following shows a successful connection test.

```
entity:    X400
test case: Connection
-----------------------------------
ss_my_mbox            : /usr/local/osi/MBXDIAGI
ss_my_ssap            : ."diags"."diags"
ss_dst_ssap           : ."MHS"."hpindon"
x4_chk_size           : 5
x4_window_size        : 1
x4_mta_name           : MTA2
x4_null_mta           : N
x4_password           : secret
-----------------------------------
osi_reg()              pipe_fd:  0x00000009   result: 0
osi_reg_cf()           pipe_fd:  0x00000009   result: 0
sbm_assoc()            shmem_id: 0x00000006   result: 0
x400_conn_encode()     result: 0
ses_con_rq()           conn_id:  0x00000001   result: 0
osi_get_prim()         pipe_fd:  0x00000009   result: 0
                       indication:  ESCONCF  (S_Connect.cnf)
ses_con_cf()           conn_id:  0x00000001   result: 0
            data:  (HEX)                                      (ASCII)
            31 80 A0 80 80 01 00 00 00 A1 80 80 01 00 81 01   1...............
            03 A2 80 A0 80 A1 80 80 04 4D 54 41 31 A1 80 16   .........MTA1...
            06 73 65 63 72 65 74 00 00 00 00 00 00 00 00 00   .secret.........
            00 00 00                                          ...
x400_accept_decode()   result: 0
ses_rel_rq()           conn_id:  0x00000001   result: 0
osi_get_prim()         pipe_fd:  0x00000009   result: 0
                       indication:  ESRELCF  (S_Release.cnf)
ses_rel_cf()           shmem_id: 0x00000006   result: 0
                       data:  NOT PRESENT
sbm_disassoc()         shmem_id: 0x00000006   result: 0
osi_shut()             pipe_fd:  0x00000009   result: 0
-----------------------------------
Received X.400 Accept PDU:
  mta name:              MTA1
  password:              secret
  checkpoint size:       5
  window size:           1
  transfer syntax:       X.409
-----------------------------------
```

```
test status:  PASSED
============================================================================
```

The first two lines indicate the test being executed. The second section of output indicates the parameters used to execute the test. The third section of output indicates the dialog which took place with the remote. The fourth section shows the contents of the connection accept received. Finally the last section indicates the test status.

Parameter Display

The meaning of the parameters shown above are described in the table at the end of this section. Some parameters are common with the Session tests. These are described in a table at the end of the Session section.

Dialog Display

As the X.400 test runs it displays its interaction with the stack and its remote peer. A description of each operation performed is given below:

- **osi_reg()** - registers this application with the local stack. No communication with the remote results from this call.
- **osi_reg_cf()** - retrieves the confirmation from the local stack regarding our registration.
- **sbm_assoc()** - allocates a segment of shared memory from the shared buffer manager (sbm). This is a local operation. Shared memory is used for data transmission in order to reduce the number of data copies performed.
- **x400_conn_encode()** - generates the RTS connect protocol data unit (PDU) which will be sent out on the session connect call.
- **ses_con_rq()** - sends a Session layer connection request to the remote node. The RTS PDU encoded by the previous operation is sent on this request.
- **osi_get_prim()** - listens for messages from the remote node. By default we will wait a maximum of 60 seconds for a message before timing out (see the Utilities section entry for "Wait Time" to adjust this value).

  If a message arrives its type is displayed on the next line of output. In this case we received a confirmation for the connection request we sent. The first item "ESCONF" is an internal name for the indication and the second item, "S_Connect.cnf", is just a readable form.
- **ses_con_cf()** - this operation decodes at the Session level the connection confirmation. Any data carried on the Session level packet is displayed. In this case the data carried is an RTS PDU.
- **x400_accept_decode()** - decodes the RTS PDU received.

- **ses_rel_rq()** - sends a request to the remote to release the Session connection.
- **osi_get_prim()** - listen for a confirmation from the remote for the release.
- **ses_rel_cf()** - decode the Session release confirmation received. Note that RTS does not send any PDU's on this Session primitive.
- **sbm_disassoc()** - release the shared memory.
- **osi_shut()** - deregister ourselves from the stack.

Decoded PDU

The remote RTS's response to our connection is displayed in the next section. The display shows that the remote did accept our connection. The MTA name and password it uses for this node is shown. The negotiated checkpoint size and window size are shown. These values may be different from the ones we proposed because they are negotiable. Finally the transfer syntax is shown, which should always appear as X.409.

Test Status

Lastly the status of the test is displayed. In this case the test passed. See the section on Interpreting Errors for example of failed test output.

Successful RTS Loopback

The following shows the results of a successful RTS loopback test. This test relies on the X.400 event logs in order to track the progress of a message being processed by the actual X.400 RTS.

```
entity:    X400
test case: RTS Loopback Test
-------------------------------------
x4_link_type        : lan
x4_loop_addr        : "HPINDWH"
-------------------------------------
 ** Running loopback over LAN **
RTS Loopback Test Started

06/12-19:56:24 Sent S-CONNECT.Request   (X4EVENT 3501)
               NSAP:TSAP:SSAP: 4850494E445748:4D4853:873:6469616773
               Outgoing MTA Name: "LANLOOPOUT"
               Outgoing Password: ""
               Checkpoint Size: 5
               Window Size: 1
               Dialogue Mode: 0
               Application Protocol: 1
               Data Transfer Syntax: 0
               Path ID: 1
06/12-19:56:24 Received S-CONNECT.Indication   (X4EVENT 3572)
               NSAP:TSAP:SSAP: 4850494E445748:4D4853:873:6469616773
               Incoming MTA Name: "LANLOOPOUT"
               Incoming Password: ""
               Checkpoint Size: 5
               Window Size: 1
               Dialogue Mode: 0
               Application Protocol: 1
               Data Transfer Syntax: 0
               Path ID: 2
06/12-19:56:25 Sent S-CONNECT.Response (Accept)   (X4EVENT 3573)
               Checkpoint Size: 5
               Window Size: 1
               Data Transfer Syntax: 0
               Path ID: 2
06/12-19:56:25 Received S-CONNECT.Confirm (Accept)   (X4EVENT 3574)
               Checkpoint Size: 5
               Window Size: 1
               Data Transfer Syntax: 0
               Path ID: 1
06/12-19:56:25 STARTED OUTBOUND TRANSFER OF 11995 BYTES
               /usr/spool/x400/looplanoq/N0.000   (X4EVENT 3502)
```

```
06/12-19:56:29 INCOMING TRANSFER OF 11995 BYTES SUCCESSFULLY COMPLETED.
               /usr/spool/x400/loopiq/M0612195626a.500  (X4EVENT 3524)
06/12-19:56:29 OUTBOUND TRANSFER SUCCESSFULLY COMPLETED.
               /usr/spool/x400/looplanoq/N0.000  (X4EVENT 3505)
06/12-19:56:29 RELEASING OUTBOUND CONNECTION.  (X4EVENT 3514)
06/12-19:56:29 INCOMING CONNECTION RELEASED.  (X4EVENT 3525)

RTS LOOPBACK TEST COMPLETED SUCCESSFULLY!
-------------------------------------
test status:  PASSED
==========================================================================
```

Successful User Agent Mail Test

The following shows a successful test of sending a mail message to ourselves. You can see the use of the various event logs to track the progress of the message.

```
entity:    X400
test case: User Agent Traced Mail Test (x4uatest)
------------------------------------
x4_surname          : root
x4_given_name       :
x4_initial          :
x4_generation       :
x4_ou1              :
x4_ou2              :
x4_ou3              :
x4_org              : SALES
x4_country          : US
x4_admd             : TELEMAIL
x4_prmd             : HP
x4_x121_addr        :
x4_ua_id            :
x4_telematic        :
x4_subject          : osidiag
x4_mail_fname       : /etc/motd
------------------------------------

  The mesage was sent through Mailx to X.400
  The Subject name is  osidiag9694
  The Recipient name is root___/__/SALES/US/TELEMAIL/HP///@
  The Originator name is root@warm
  The Filename sent is /etc/motd

 The Internal Filename sent by x4mailer to MTA was
  /usr/spool/x400/iq/I0612182202a.000

*******************************************************
             TRACKING OUTGOING MESSAGE
*******************************************************

>> Analyzing /usr/spool/x400/log/x4mailout.evnt log file for
   MPDUID /usr/spool/x400/iq/I06121
============================================
=== Message found in X4MAILOUT event log ===
============================================

  -- Start of event 414 --
  06/12-18:22:02 UMPDU submitted to MTA : (X4EVENT 414)
                  MPDU File [/usr/spool/x400/iq/I0612182202a.000]
```

```
                 Mpdu-id : US TELEMAIL HP            F102253   645240121B

>> Analyzing /usr/spool/x400/log/mta.evnt log file for I0612182202a filename <<

========================================
=== Message found in MTA event log  ===
========================================

  -- Start of event 2114 --
  06/12-18:22:05 UMPDU 0 ARRIVED in file
                 /usr/spool/x400/iq/I0612182202a.000  (X4EVENT 2114)
  Mpdu-id : US TELEMAIL HP WARM        F102253   645240121B
  Originator : CNTRY=US ADMD=TELEMAIL PRMD=HP ORG=SALES SUR=root
  Filesize : 589  bytes
  Priority  :  0
  Content Type : 2

  -- Start of event 2108 --
  06/12-18:22:06 Internal Trace entry 0 from warm.x4mailer
  Arrived at Jun 12 18:22:01 1990 PDT  Action = RELAYED

  06/12-18:22:07 UMPDU 1 RELAYED to MTA 253 named WARM in file
                 /usr/spool/x400/oq/oq253/N0612182206a.253.  (X4EVENT 2108)
  Mpdu-id : US TELEMAIL HP WARM        F102253   645240121B
  Originator : CNTRY=US ADMD=TELEMAIL PRMD=HP ORG=SALES SUR=root
  Recipient : CNTRY=us ADMD=telemail PRMD=hp ORG=sales SUR=root
  PerRecipientFlag : d0(Hex), 208(Decimal)

  ************************************************************************

  -- Start of event 2114 --
  06/12-18:22:47 DR-MPDU 0 ARRIVED in file
                 /usr/spool/x400/iq/I0612182239a.000  (X4EVENT 2114)
  Mpdu-id : US TELEMAIL HP WARM        F102253   645240159
  Original Mpdu-id : US TELEMAIL HP WARM        F102253   645240121B
  Originator : CNTRY=US ADMD=TELEMAIL PRMD=HP ORG=SALES SUR=root
  Filesize : 503  bytes

  -- Start of event 2108 --
  06/12-18:22:47 Internal Trace entry 0 from warm.x4mailer
  Arrived at Jun 12 18:22:39 1990 PDT  Action = RELAYED

  06/12-18:22:48 DR-MPDU 0 RELAYED to MTA 253 named WARM in file
                 /usr/spool/x400/oq/oq253/N0612182248a.253.  (X4EVENT 2108)
  Mpdu-id : US TELEMAIL HP WARM        F102253   645240159
  Original Mpdu-id : US TELEMAIL HP WARM        F102253   645240121B
  Originator : CNTRY=US ADMD=TELEMAIL PRMD=HP ORG=SALES SUR=root
```

```
* 3 possible message path(s) found in MTA *
* Checking message path 1 of 3 *


>> Analyzing /usr/spool/x400/log/x4mailin.evnt log file for N0612182206a filename <<

=====================================
=== Message found in X4MAILIN log ===
=====================================

  -- Start of event 402 --
  06/12-18:22:10 Processing UserMPDU  (X4EVENT 402)
                 MPDU File [/usr/spool/x400/oq/oq253/N0612182206a.253]
  Mpdu-id : US TELEMAIL HP WARM       F102253   645240121B
  Originator : root___//SALES/US/TELEMAIL/HP////
  Recipient : root@warm

* Checking message path 2 of 3 *


>> Analyzing /usr/spool/x400/log/x4mailin.evnt log file for I0612182239a filename <<

=====================================
=== Message found in X4MAILIN log ===
=====================================

  -- Start of event 414 --
  06/12-18:22:40 DR-MPDU submitted to MTA : (X4EVENT 414)
                 MPDU File [/usr/spool/x400/iq/I0612182239a.000]
                 Mpdu-id : US TELEMAIL HP             F102253   645240159

* Checking message path 3 of 3 *


>> Analyzing /usr/spool/x400/log/x4mailin.evnt log file for N0612182248a filename <<

=====================================
=== Message found in X4MAILIN log ===
=====================================

  -- Start of event 403 --
  06/12-18:22:52 Processing DeliveryReportMPDU  (X4EVENT 403)
                 MPDU File [/usr/spool/x400/oq/oq253/N0612182248a.253]
  Originator : root___//SALES/US/TELEMAIL/HP////
  Recipient : root___//sales/us/telemail/hp////
  Mpdu-id : US TELEMAIL HP WARM       F102253   645240121B
```

```
**********************************************************
               COMPLETED TRACKING  OUTGOING MESSAGE
**********************************************************

**********************************************************
               TRACKING INCOMING DELIVERY REPORT
**********************************************************


******************************************
            The OUTPUT FROM  MTA log    *
******************************************

06/12-18:22:48 DR-MPDU 0 RELAYED to MTA 253 named WARM in file
                   /usr/spool/x400/oq/oq253/N0612182248a.253.  (X4EVENT 2108)
Mpdu-id : US TELEMAIL HP WARM        F102253   645240159
Original Mpdu-id : US TELEMAIL HP WARM        F102253   645240121B

*****************************************
       Results from DELIVERY report    *
*****************************************

****** DELIVERY REPORT ARRIVED  *****

06/12-18:22:52 Processing DeliveryReportMPDU  (X4EVENT 403)
                   MPDU File [/usr/spool/x400/oq/oq253/N0612182248a.253]
Originator : root___//SALES/US/TELEMAIL/HP////
Recipient : root___//sales/us/telemail/hp////
Mpdu-id : US TELEMAIL HP WARM        F102253   645240121B

-------------------------------------
test status:  PASSED
======================================================================
```

Interpreting Results

In the above example we sent a loopback mail message to ourselves. From the queue model shown in the diagram earlier in this section the message will go into the MTA's input queue, and the MTA will route it immediately back to the X4mailing input queue oq253.

The various events are described below:

- TRACKING OUTGOING MESSAGE - this indicates that we are beginning to examine the event logs for messages showing the progress of our mail message.
- 06/12-18:22:02 UMPDU submitted to MTA : (X4EVENT 414) - this indicates that x4mailer has passed a message to the MTA.

- 06/12-18:22:05 UMPDU 0 ARRIVED in file - this indicates the MTA's reciept of the message from x4mailer.
- 06/12-18:22:07 UMPDU 1 RELAYED to MTA 253 named WARM in file - this indicates that the MTA knows where to send this message and is about to relay it. The MTA name is given. In this case it is a loopback, so it goes to MTA 253. If it were not loopback a number like MTA 1 would be given and its name.
- 06/12-18:22:40 DR-MPDU submitted to MTA : (X4EVENT 414) - this indicates that the x4mailer has received the loopback message and is sending a delivery notification to the sender.
- COMPLETED TRACKING OUTGOING MESSAGE - this indicates that the message transfer stage has completed and we are now awaiting the delivery notification.
- 06/12-18:22:52 Processing DeliveryReportMPDU (X4EVENT 403) - this indicates that we have received the delivery report and gives further details about the delivery status.

Interpreting Errors

If a X.400 test operation fails, the following output will appear in the status section at the end of the output:

```
------------------------------------
test status:  FAILED
operation:    setting up test
return code:  0
osierrno:     0
meaning:      No additional information.


x4uatest: Error detected performing User Agent Test.
          Examine the test report above for further information.
==========================================================================
```

The meaning of each field is as follows:

- **test status** - indicates whether we passed or failed. Any error encountered during the operation (including setting up tracing and logging, if requested) will result in a status of failed.
- **operation** - indicates at what point in the test we encountered the first error. For the RTS loopback and User Agent mail tests this will typically be "setting up test". For the simulated tests this will typically be a Session interface call.
- **return code** - this is the return value from a Session interface call (if any).

  In some instances (as the one shown above) this value will contain an osidiag specfic error. See the section in this Appendix on Session for more information.
- **osierrno, meaning** - this contains any additional error information from the Session interface and what that value means. A value of zero indicates no additional information was given.
- **Formatted error text** - after the above fields are displayed, osidiag will provide any textual information about the error. Typically this is displaying the meaning of the return code. In this case "x4uatest: Error detected performing User Agent Test."

Further Information

Further information on interpreting errors may be found in the Troubleshooting section of this manual. Also check the X.400 product manauls.

Parameter Summary

The following table describes the parameters used for X.400 testing. Those parameters beginning with the prefix "ss_" are the same as for the Session tests.

For a description of the valid syntax for the various parameters see the section titled "Parameter Syntaxes".

Some of these parameters have default values which you will not be prompted for. To override these defaults, turn on prompting for the appropriate class of parameter. See the subsection "Prompt Level" under "Utilities" for more information. Note: the window size and checkpoint size are considered data on connection prompts.

X.400 Parameters

parameter	syntax	description
ss_my_ssap	local address	Specifies the local Session and Transport selectors which will be used to connect through or receive a connection for the simulated RTS tests.
ss_dst_ssap	ssap	Specifies the SSAP which we will attempt to connect to. There should already be a process listening on this specified address. For simulated RTS tests only.
ss_my_mbox	string	Specifies the pipe file which we will use for message passing between osidiag and the local OTS stack. For simulated RTS tests only.
x4_chk_size	integer	Specifies the checkpoint size to be negotiated for the connection.
x4_window_size	integer	Specifies the window size to be negotiated for the connection.
x4_mta_name	string	Specifies the name that our local node is known to the remote as.
x4_password	string	Specifies the password associated with our MTA name.
x4_null_mta	yes_no	Specifies the whether any MTA naming information will be carried.

X.400 Parameters (continued)		
parameter	syntax	description
x4_orig_id	string	Specifies originator field to be filled in on a mail message which is sent for the User Agent mail test. This is only used if x4_prod_name is "H".
x4_prod_name	string	Specifies product to be tested for the User Agent mail test. The mail message may either be sent via mailx, "M", or it may be sent via the HPDesk gateway, "H".
x4_surname	string	the surname component of the X.400 recipient for the User Agent mail test.
x4_given_name	string	the given name component of the X.400 recipient.
x4_initial	string	any initials for the X.400 recipient.
x4_generation	string	any generational qualifier for the X.400 recipient.
x4_ou1,ou2,ou3	string	orginizational unit components of the X.400 recipient.
x4_org	string	the orginization name component of the X.400 recipient.
x4_country	string	the country code component of the X.400 recipient.
x4_admd	string	the administrative management domain component of the X.400 recipient.
x4_prmd	string	the private management domain component of the X.400 recipient.
x4_x121_addr	string	the X.121 address component of the X.400 recipient. Usually not present.
x4_ua_id	string	the user agent identifier component of the X.400 recipient. Usually not present.
x4_telematic	string	the teletex identifier component of the X.400 recipient. Usually not present.

X.400 Parameters (continued)		
parameter	**syntax**	**description**
x4_subject	string	This is the subject text which will be provided for the User Agent mail test.
x4_mail_fname	string	This is the name of a file which contains the test message which will be sent for the User Agent mail test.
x4_link_type	string	This is the link which the RTS loopback test will exercise. This should have a value of either "lan" or "wan".
x4_loop_addr	octet string	This is the network address which will be used for the RTS loopback test.

CMIS Tests

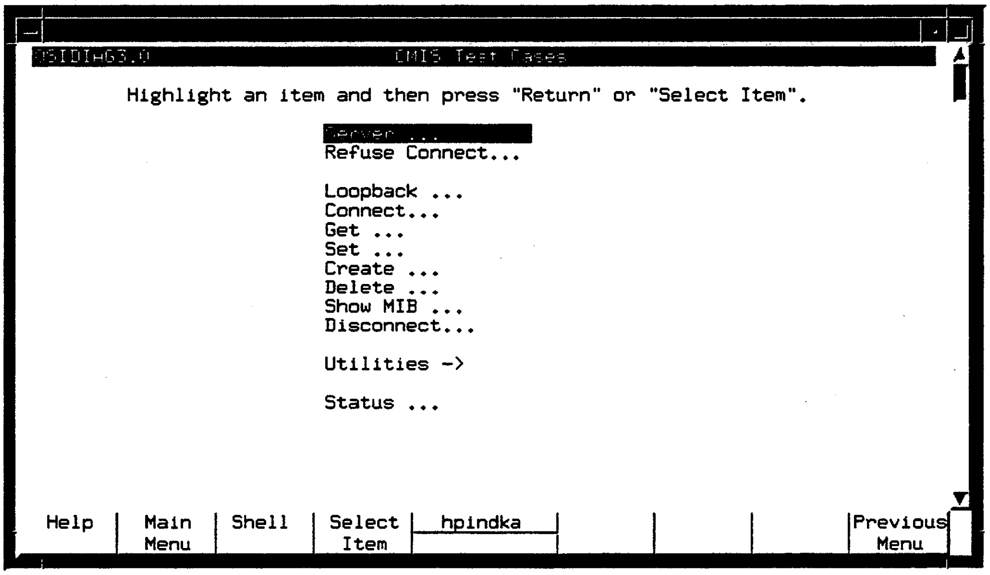
Osidiag allows you to generate and receive CMIP primitives through the CMIP API. Specifically osidiag will allow the user to create or accept connections, to create or delete managed objects, and to get and set the attributes of a particular managed object.

In addition to providing manager functionality, osidiag will provide access to a user definable Management Information Base (MIB), which a remote entity can access via the Create, Delete, Get and Set services. The MIB provided by osidiag will not map to any real managed resources, it will be purely for the purposes of interoperability testing.

NOTE: Osidiag will attempt to use the CLNS service through CMIS by default. In order to use CONS (for X.25) see the description of Local SAP below.

Test Menu

The test case menu displayed for the CMIS tests is shown below:



```
OSIDIAG3.0                        CMIS Test Cases

        Highlight an item and then press "Return" or "Select Item".

                          Server ...
                          Refuse Connect...

                          Loopback ...
                          Connect...
                          Get ...
                          Set ...
                          Create ...
                          Delete ...
                          Show MIB ...
                          Disconnect...

                          Utilities ->

                          Status ...
```

Server

This operation results in osidiag waiting for an inbound event from the CMIP interface. If the event is a connection indication, osidiag will respond with a confirmation. If the event is a Get, Set, Create, or Delete, osidiag will attempt to satisfy the request by examining its MIB. If the request is acceptable, osidiag will provide a response (when necessary). If the request is invalid, then osidiag will issue an Error response.

By default, once the server operation is invoked, it will continue to receive and respond to inbound events until an associate release is received. At that time osidiag will respond to the release and return control to the user. Alternatively the user may chose to have osidiag listen (and respond to) a single primitive and then return control. This is controlled via the "Save Connections" facility under "Utilities".

Refuse Connect

This operation results in osidiag listening for an inbound connection and then rejecting that connection.

Connect

This operation results in osidiag issuing an associate request to the remote entity. By default osidiag will immediately release the connection after the the confirmation is received.

Alternatively by selecting "Save Connections" under the "Utilities" menu, osidiag will return control to the user after receiving the confirmation, without releasing it. Allowing further CMIP operations to be performed over this connection (i.e. Get, Set, etc.).

Loopback

This operation causes osidiag to issue a connect request to itself, receive and respond to the connection, and then release it. "Save Connections" is not available for loopback.

This operation is useful for doing a quick verification that all the local facilities are available for CMIP network activity.

Disconnect

This operation is used to release a connection which has been left open as a result of choosing "Save Connections".

Get

This operation allows the user to get one or more attributes from a specific managed object instance. If successful the attribute values returned will be displayed. If unsuccessful, the error response received will be displayed.

Set

This operation allows the user to set one or more attributes of a specific managed object instance. If unsuccessful, the error response received will be displayed.

Create

This operation allows the user to create a new object instance on the remote.

Delete

This operation allows the user to remove an object instance on the remote.

Show MIB

This operation will display the current contents of osidiag's MIB. This will reflect the values originally specified by the user (or the defaults) in combination with any modifications made to the MIB via Set, Create, or Delete operations issued by the remote.

Utilities

By selecting this item you will be taken to the Utilities menu. To return from the Utilities to the X.25 tests press Exit.

Status

This will invoke "cmistat" to display the status of CMIS link and "osiconfchk" to display configuration information.

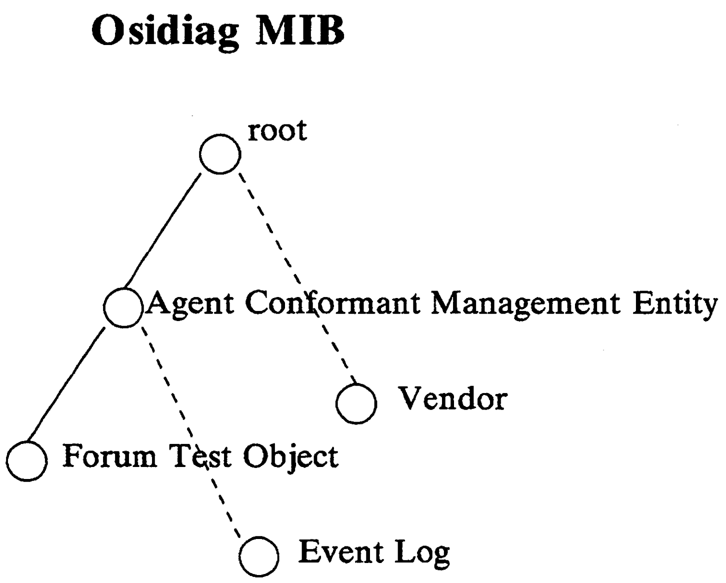
Osidiag Limitations

The following lists CMIP features which osidiag does not support.

- Action - The CMIP action service is not supported.
- Event Report - This CMIP service is supported as an inbound indication, however osidiag will not generate event reports.
- Scope/Filter - Object scoping and filtering for performing actions on multiple object instances is not supported.
- Local Distinguished Names - Only Distinguished Names and Local (Object Id) references are allowed for inbound or outbound calls.
- Session Connect Id - This parameter is not exposed to the user and will always be left out of connection calls.
- AP Titles/AE Qualifiers - These parameters are not exposed to the user and will always be left out of connection calls.
- Quality of Service - This parameter will not be exposed to the user.

Osidiag's MIB

The figure below shows the default MIB created by osidiag. The dashed components are not part of the default MIB and are shown only to emphasize the tree structure of the MIB.

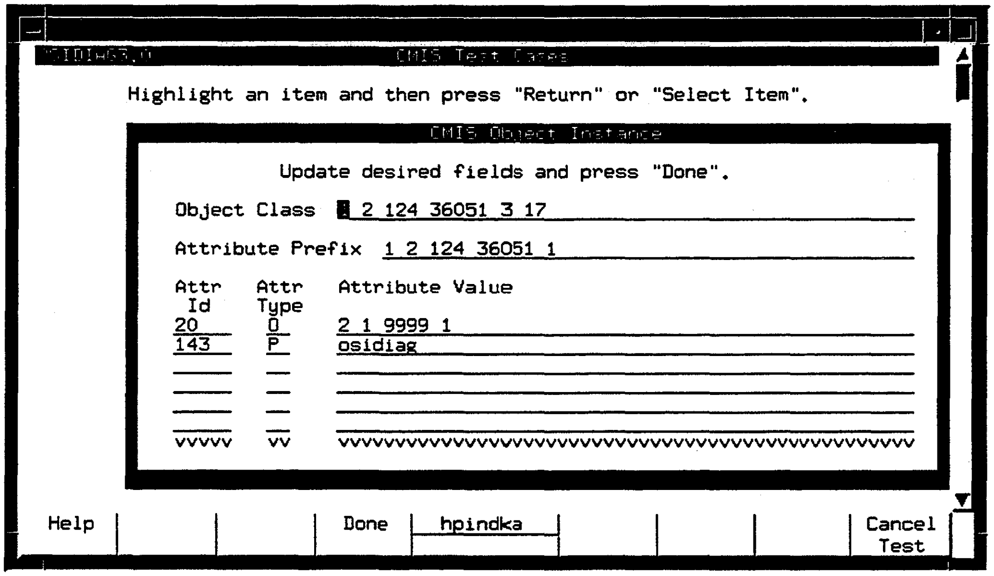


Note that osidiag by default will contain only two object instances. Both object instances are as defined by NM Forum. The user may create other initial MIB structures by providing a MIB definition in an osidiag parameter file.

Parameters

CMIS Object Instance

The figure below shows the osidiag "object instance" parameter screen.



```
OSIDIAG3.0                    CMIS Test Cases

      Highlight an item and then press "Return" or "Select Item".

                         CMIS Object Instance

              Update desired fields and press "Done".
        Object Class  █ 2 124 36051 3 17
        Attribute Prefix  1 2 124 36051 1
        Attr   Attr   Attribute Value
         Id    Type
        20      O     2 1 9999 1
        143     P     osidiag
        vvvvv   vv    vvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvv
```

This window is presented for the operations, Get, Set, Create and Delete. It is used to locate an object to be manipulated in the remote's MIB. The data entered maps onto several CMIP API parameters.

The default value allows you to access the "Forum Test Object" object instance in osidiag's default MIB.

The Parameter components are as follows.

- **Object Class** - This describes the object instance being located. It can be either an object identifier (as shown in the example) or it may be an integer.
- **Attribute Prefix** - This value is the beginning sequence of the object id's which describe the attributes which make up the distinguished name. It presumes (as in NM Forum) that all attributes have a common object id with the exception of the last number. It is combined with each "Attr Id" parameter to construct the actual object id which is passed to the CMIP API.

- **Attr Id** - This value (combined with the "Attribute Prefix") describes the attribute which is being used to traverse the MIB tree to find the object instance being referenced. If the remote uses the "local" form to describe attribute identifiers, then the "Attribute Prefix" field should be left blank, and the integer value specified will be passed as a local identifier.
- **Attr Type** - This describes the type of the attribute value. Because the attributes may be of any form, osidiag provides the following attribute types:
    - O "object identifier," the value provided in the "Attribute Value" field should follow the syntax shown in the example.
    - I "integer," the value should be a decimal integer value.
    - R "raw hex," the value should be a sequence of hex digits representing an arbitrary octet string. This value will be passed literally, and hence should follow the BER. (i.e. 30 06 02 01 12 02 01 E7).
    - P "printable," the value will be passed as an ASN.1 printable string. (i.e Hello World).
    - F "file," this is similar to "raw hex" mode, except the raw data to be passed is contained in a file. This is useful for large BER encoded structures. The argument is a file name (i.e. /tmp/value).
- **Attribute Value** This value is dependent on the type selected. It describes the value of the attribute which must be matched to succesfully locate the object. See the discussion above of "Attribute Type" for discussion and examples of the various values.

Remote Address

Osidiag will prompt for the remote address with which to connect. It has the same form as described for the FTAM tests.

By default it will contain "cmid".<SSel>.<TSel>.<local_addr>. Where <SSel>, <TSel> and <local_addr> are extracted from the local configuration. The user will always be prompted for this value before establishing connections.

Local SAP

CMIS allows applications to bind by address and type, where address is a P-, S- and T-selector and type specifies CLNS or CONS. Osidiag will specify a type code indicating CLNS by default.

In order to use CONS, you must change the parameter cmis_my_sap to the value "-2". The value "-1" indicates CLNS. A positive value is assumed to be a valid SAP number (as discussed in the CMIS Programmer's Guide). If a positive value is given osidiag will bind by SAP rather than by address.

Osidiag will not prompt for this parameter by default, so if you want to use a non-default value, you need to modify the "Prompt Level" flags under the "Utilities" menu. Alternatively you can use a parameter file to store the non-default value.

Application Context Name

The context which will be used for the association with the remote. A default will be provided which the user may override at connection setup time. The screen is similar to that described in the APLI section.

CMIS Object Attribute Identifiers

When running the Get operation, this list specifies a specific set of attributes to be retrieved from the specified object instance. If left blank, all attributes for the object instance will be retrieved.

The Object Attribute Ids screen has two fields, an Attribute Prefix and an Attribute Id list. These are used the same way as described above for CMIS Object Instance.

CMIS Object Attributes

When running the Set or Create operation, this list specifies values that an attribute or attributes of the object instance are to take on. This field should not be left blank.

The Object Attributes screen has four fields, an Attribute Prefix and an Attribute Id list, an Attribute Type list and an Attribute value list. These are used the same way as described above for CMIS Object Instance.

CMIS Parameters

The following table describes the parameters used for the CMIS interface.

For a description of the valid syntax for the various parameters see the subsection titled "Parameter Syntaxes" later in this chapter.

Some of these parameters have default values which you will not be prompted for. To override these defaults, turn on prompting for the appropriate class of parameter. See the subsection "Prompt Level" under "Utilities" for more information.

CMIS Parameters		
parameter	**syntax**	**description**
cmis_my_psap	paddr	Specifies the local Presentation Address which we will either establish our connection through or receive a connection.
cmis_dst_psap	paddr	Specifies the Presentation address which we will attempt to connect to. There should already be a process listening on this specified address.
cmis_appl_ctx	objid	Specifies the application context to be used for this connection.
cmis_my_sap	integer	Specifies the SAP number to bind on. If -1, then bind by address CLNS. If -2, then bind by address CONS. If > 0 then bind by SAP.
cmis_obj_instance	attr_val	Specifies an object instance to be gotten or set by giving a distinguished name. This name indicates the point in the MIB where the object instance is located.
cmis_obj_class	refid	Defines the object class of the object instance we are operating on.
cmis_mib	mib	Specifies the contents of the MIB which osidiag will make avaiable when running server operation.
cmis_attr_id_list	attr_id	This list gives the attributes to be retrieved during a Get operation. If blank, all attributes retrieved.
cmis_attr_list	attr_val	This list specifies attributes and values to used for a Set or Create operation.

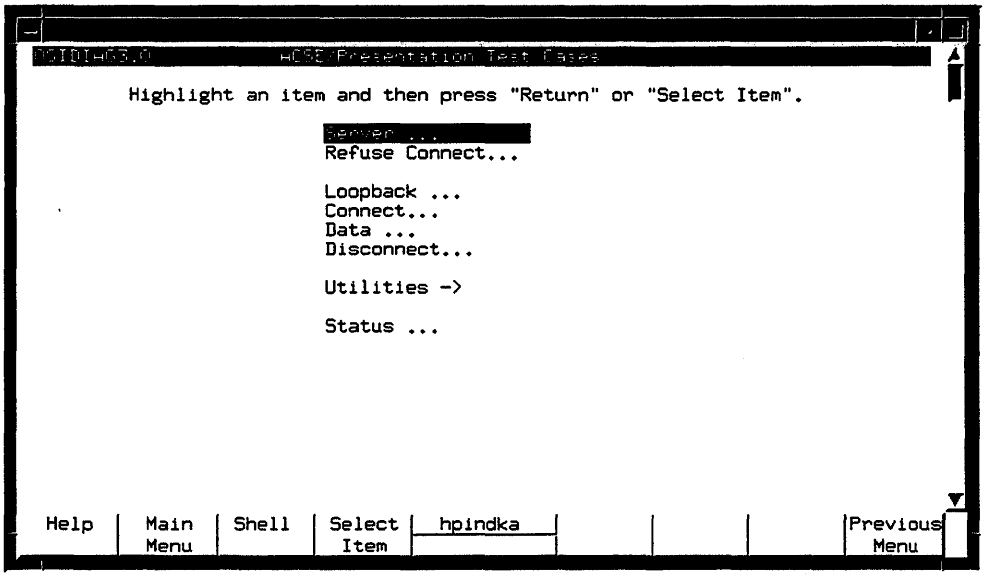
CMIS Parameters (cont.)		
parameter	**syntax**	**description**
cmis_ref_reason	integer	This is the refuse code passed on the CMIS Refuse Connection test. It can only be changed via parameter files. Default value 0.
cmis_invoke_id	integer	This is the invoke id used on CMIS requests. It can only be changed via parameter files. Default value 1.
cmis_p_ctx_list	pcdl	NOT USED.

APLI Tests

Osidiag allows you to verify the ability to connect to and receive connections from a remote node through the ACSE/Presentation Layer Interface. Also provided is the ability to send data to the remote over a ACSE/Presentation connection.

Test Menu

The test case menu displayed for the APLI tests is shown below:



```
OSIDIAG3.0                ACSE/Presentation Test Cases

      Highlight an item and then press "Return" or "Select Item".

                         Server ...
                         Refuse Connect...

                         Loopback ...
                         Connect...
                         Data ...
                         Disconnect...

                         Utilities ->

                         Status ...
```

Server

The server test will listen for an inbound connection and accept it. After accepting the connection the test will listen for other APLI indications such as data or disconnect. Indications other than the disconnect will be reported and osidiag will continue to listen. When a disconnect is received the test ends.

The test will wait for a default duration of 30 seconds. If after 30 seconds no inbound connection is received the test will terminate with a timeout failure. The timeout duration can be modified by going to the utilities menu and selecting "Wait Time" prior to the execution of the test. This timeout also holds for subsequent listens for data and disconnect indications.

This test can also be used to receive single events and then return control to the menu. See the subsection "Save Connections" under Utilities.

Refuse Connect

This test is similar to the Server test in that it will wait for an incoming connection indication. The difference is that rather than accepting the connection it will return an refuse.

Connect

The connect test will result in a connection request being sent to the remote ACSE/Presentation peer. After the connection is confirmed it is dissolved by sending a release.

This test can also be used to establish and leave open a connection. See the subsection "Save Connections" under Utilities.

By default the connection will go out only over the configured CLNS subnetwork. This determination is based on the network address used for the local Presentation Address (parameter ap_my_psap). You should run a loopback test over CONS, or change the prompt level under Utilities (to include Local SAP) in order to run this test over the non-default subnetwork.

Loopback

The loopback test will perform the functions of the server and connect tests in one test. A listen will be posted for an incoming connection and then we will issue a connection to ourselves. After the connection has been successfully established it will be torn down.

When running this test you will be prompted for the destination address. If you only have one link configured then the default will suffice. If you have both LAN and X.25 links configured the default will correspond to LAN. To test over X.25 in this case, change the network address to that configured for CONS under osiconf.

Data

This test will send a Presnetation data PDU over an APLI connection. If no connection currently exists, one will be established. The user is prompted for the data to be carried in the PDU.

Disconnect

This test is only useful when saving connections across tests (as described in the "Save Connections" subsection of the Utilities section). It will shut down the existing connection and the registration with the stack.

Utilities

By selecting this item you will be taken to the Utilities menu. To return from the Utilities to the Transport tests press Exit.

Status

This operation invokes the program otsstat(1M) to display the current status of OTS. It also invokes osiconfchk(1M) to display a summary of important configured values (e.g. local addresses, destination systems, etc.).

Parameters

The following describes the parameters which osidiag will request during the various tests.

Remote APRI Presentation Address

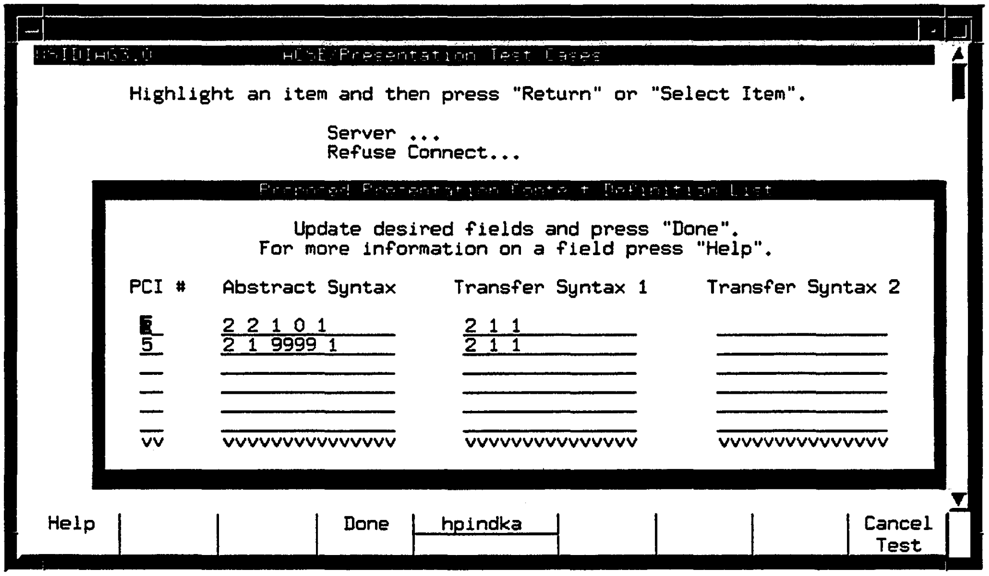
When running any test which requires a connection to be established a window is brought up for you to enter the address of the remote entity which we will attempt to connect. The window is similar to that described for FTAM in this chapter.

The default value presented by osidiag is "aprd"."aprd"."aprd".<local net addr>. Where local net addr is the configured local network address. The When running the Loopback test or the Connect and Server on the same node this will be sufficient.

When running against a remote HP node also running osidiag, you need only change the Network Address. For non-HP nodes this value should be obtained by asking the network administrator for the remote node.

Proposed Presentation Context Definition List

When establishing a connection you will be prompted for the a list of Presentation Contexts to be used for the life of the connection. The following form is presented:



```
OSIDIAG3.0              ACSE/Presentation Test Cases

        Highlight an item and then press "Return" or "Select Item".

                        Server ...
                        Refuse Connect...

             Proposed Presentation Context Definition List

                    Update desired fields and press "Done".
                  For more information on a field press "Help".

        PCI #   Abstract Syntax     Transfer Syntax 1    Transfer Syntax 2

        1_      2 2 1 0 1_______    2 1 1___________     ________________
        5_      2 1 9999 1______    2 1 1___________     ________________
        __      ________________    ________________     ________________
        __      ________________    ________________     ________________
        __      ________________    ________________     ________________
        __      ________________    ________________     ________________
        vv      vvvvvvvvvvvvvvvv    vvvvvvvvvvvvvvvv     vvvvvvvvvvvvvvvv

  Help  |     |     |  Done  | hpindka |     |     |  Cancel
                                                       Test
```

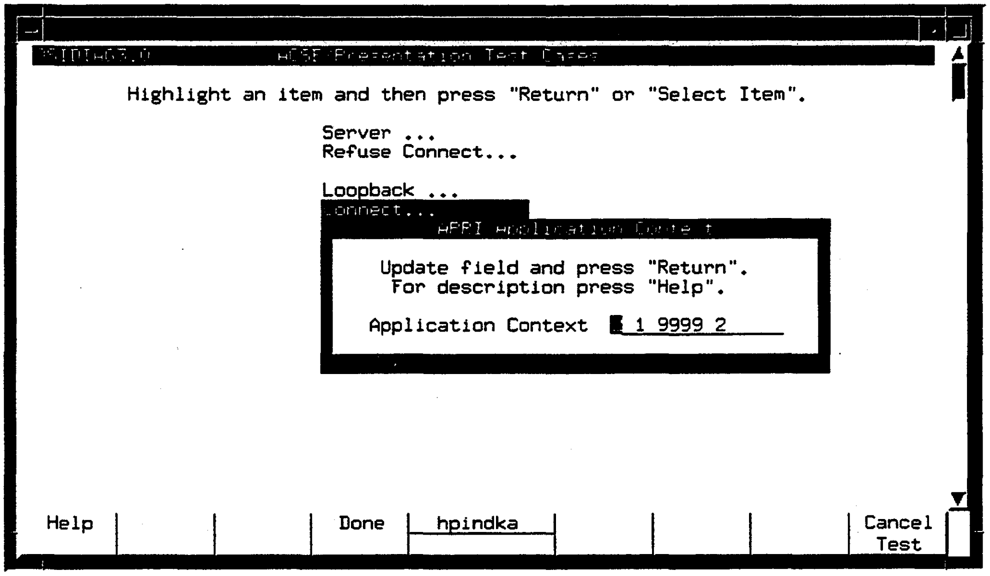
Presentation contexts indicate what protocols are to be used in addition to Presentation. So typically there are at least two contexts proposed, that for ACSE, and one or more for the application which is running over APLI.

The proposed values are as follows:

- **PCI** - this is an integer which will be used as an id number for future references to this context.
- **Abstract Syntax** - this is an object id which identifies a protocol. These are defined by standards such as FTAM, MMS, ACSE. For user developed applications which use their own protocol over APLI, a unique value should be created.
- **Transfer Syntax(es)** - these are object id's which describe how the protocol indicated by the Abstract Syntax is encoded. This will typically be the object id for ASN.1, {2 1 1}. Multiple syntaxes may be proposed, but there are no other widely used transfer syntaxes at this writing.

APRI Application Context

When establlishing a connection you must also specify the application context. You will be give the following pop up window.



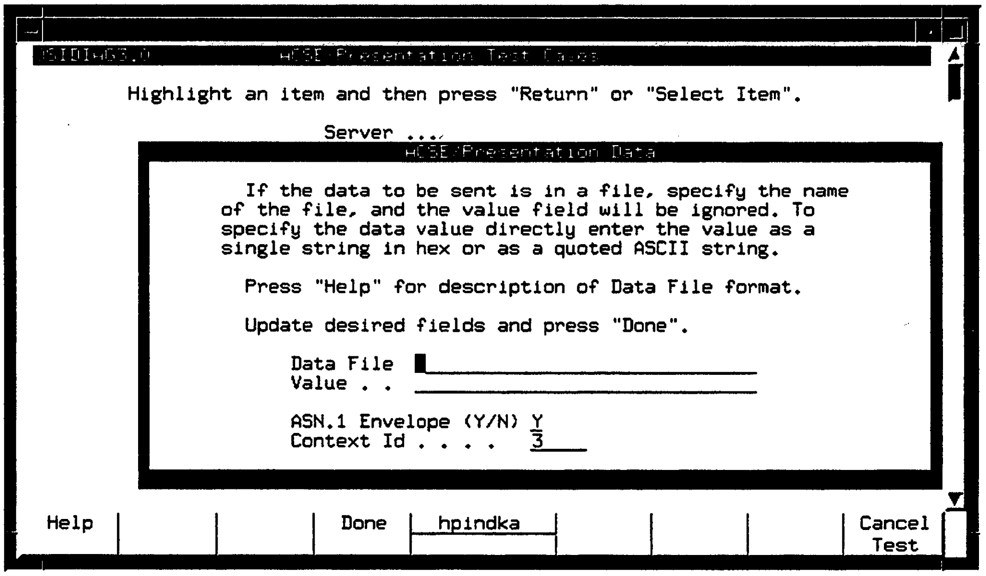
```
OSIDIAG3.0            ACSE/Presentation Test Cases

      Highlight an item and then press "Return" or "Select Item".

                        Server ...
                        Refuse Connect...

                        Loopback ...
                        Connect...
                                APRI Application Context

                          Update field and press "Return".
                           For description press "Help".

                         Application Context  1 9999 2
```

This field specifies the application context to be used over this ACSE connection. The context provides a mechanism for identifying the type of application which is requesting the connection.

Services such as FTAM and MMS define standard application contexts which must be used. For user developed layer 7 applications, the developer must decide what value to use.

ACSE/Presentation Data

When transfering data you will be prompted with the following window.



```
OSIDIAG3.0                ACSE/Presentation Test Cases

          Highlight an item and then press "Return" or "Select Item".

                          Server ...
                          ACSE/Presentation Data

                 If the data to be sent is in a file, specify the name
               of the file, and the value field will be ignored. To
               specify the data value directly enter the value as a
               single string in hex or as a quoted ASCII string.

                 Press "Help" for description of Data File format.

                 Update desired fields and press "Done".

                   Data File  ______________________
                   Value . .  ______________________

                   ASN.1 Envelope (Y/N) Y
                   Context Id . . . .   3___
```

The "Data File" and "Value" fields behave as describe in the Session Tests discussion.

The field "ASN.1 Envelope" indicates whether osidiag should enclose the data value specifed with ASN.1 headers, to guarantee that it is complies with the ASN.1 encoding rules.

The following ASN.1 structure is used for data on connect or release:

```
[30] IMPLICIT SEQUENCE OF EXTERNAL {
     pci INTEGER;
     [1] UserData;
}
```

This structure is used for data carried on a P_DATA_REQ:

```
[APPL 1] IMPLICIT SEQUENCE OF PDV-list{
     pci INTEGER;
     [1] UserData;
}
```

If you specify "N", then you must ensure that the data you enter is ASN.1 encoded or an error may result.

The Context Id field gives the PCI value which will be used if ASN.1 Envelope is set to "Y". It should match one of the negotiated PCI value.

Note if the Envelope Flag is set to "Y" then the ASN.1 header will be sent carrying any user data present in the Data File or Value fields. If no data is specified an empty ASN.1 header will still be sent.

Example

The following shows a successful connection test.

```
entity:    APLI
test case: Connection
------------------------------------
ap_my_psap           : "aprd"."aprd"."aprd"."hpindka"
ap_dst_psap          : "aprd"."aprd"."aprd"."hpindka"
ap_p_ctx_list        : {{3 {2 2 1 0 1}{{2 1 1}}}{5 {2 1 9999 1}{{2 1 1}}}}
ap_appl_ctx          : 2 1 9999 2
ap_cdata_val         :
ap_cdata_file        :
ap_pci               : 3
ap_envelope          : Y
ap_rdata_val         :
ap_rdata_file        :
ap_pci               : 3
ap_envelope          : Y
------------------------------------
ap_open()             fd: 09   result: 0  ap_errno: 0x0
ap_init_env()         fd: 09   result: 0  ap_errno: 0x0
ap_snd(A_ASSOC_REQ)   fd: 09   result: 0  ap_errno: 0x0
ap_poll()             fd: 09   result: 0  ap_errno: 0x0
ap_rcv()              fd: 09   result: 0  ap_errno: 0x0
                        indication:  A_ASSOC_CNF
            result:  AP_ACCEPT
            data:  (HEX)                                  (ASCII)
            BE 80 28 05 02 01 03 81 00 00 00              ..(........
ap_snd(A_RELEASE_REQ)  fd: 09   result: 0  ap_errno: 0x0
ap_poll()             fd: 09   result: 0  ap_errno: 0x0
ap_rcv()              fd: 09   result: 0  ap_errno: 0x0
                        indication:  A_RELEASE_CNF
            data:  (HEX)                                  (ASCII)
            BE 80 28 05 02 01 03 81 00 00 00              ..(........
ap_close()            fd: 09   result: 0  ap_errno: 0x0
------------------------------------
------------------------------------
test status:  PASSED
=======================================================================
```

The first two lines indicate the test being executed. The second section of output indicates the parameters used to execute the test. The third section of output indicates the dialog which took place with the remote. The fourth section shows the contents of the connection accept received. Finally the last section indicates the test status.

Parameter Display

The meaning of the parameters is given in the table at the end of this section.

Dialog Display

As the APLI test runs it displays its interaction with the interface and its remote peer. A description of each operation performed is given below:

- **ap_open()** - creates an APLI endpoint (a file descriptor) to be used for our APLI communication. No remote dialog results from this call.
- **ap_init_env()** - initializes the APLI environment. Various attributes in this environments (such as the local and remote address) are subsequently modified by osidiag via the ap_set_env() call. These calls are not displayed during the run, but will be reported if an error occurs.
- **ap_snd(A_ASSOC_REQ)** - All APLI primitives are sent via the same function, ap_snd(). An argument to this function indicates what primitive is to be sent. In this case osidiag is sending an Association Request, in an attempt to establish a connection with the remote.
- **ap_poll()** - listens for messages from the remote node. By default we will wait a maximum of 30 seconds for a message before timing out (see the Utilities section entry for "Wait Time" to adjust this value).

  If a message arrives osidiag will then call ap_rcv() to receive the primtive.
- **ap_rcv()** - this operation is used to receive all APLI indications and confirmations. Upon successful completion, the indication code is displayed, followed by any data carried on that indication. Note that the data will typically be ASN.1 encoded, as in the example above.
- **ap_snd(A_RELEASE_REQ)** - sends a request to the remote to disconnect.
- **ap_poll()** - listen for the release confirmation.
- **ap_rcv()** - receive the release confirmation.
- **ap_close()** - close the file descriptor.

Negotiated Parameters

Osidiag displays no negotiated values for APLI.

Test Status

Lastly the status of the test is displayed. In this case the test passed. See the next section for information on failed tests.

Interpreting Errors

If an APLI test operation fails, the following output will appear in the status section at the end of the output:

```
------------------------------------
test status:  FAILED
operation:    ap_set_env()
return code:  134217738


APLI Error (0x800000A):  Bad value for environment attribute

Additional Info: fd = 6,  AP_STATE = AP_UNBOUND

osidiag: Bad local presentation address. (BT604)
     The value specified for your local presentation address was rejected
     by APLI when attempting to set it via the ap_set_env(AP_BIND_PADDR)
     command.  Verify that the Network Address field is locally configured.
```

The meaning of each field is as follows:

- **test status** - indicates whether we passed or failed. Any error encountered during the operation (including setting up tracing and logging, if requested) will result in a status of failed.
- **operation** - indicates at what point in the test we encountered the first error. Typically this will be the name of an APLI operation. If an error was detected, but no APLI call failed, then the operation may say "setting up test".
- **return code** - this is the value of the APLI error returned. The meaning of this error (and its representaion in hex is given on the following line).

- **Formatted error text** - after the above fields are displayed, osidiag will provide any textual information about the error. Typically this is displaying the meaning of the return code.

Further Information

Further information on interpreting errors may be found in the Troubleshooting chapter of this manual. Also check the APRI Programmer's manual.

APLI Parameters

The following table describes the parameters used for the ACSE/Presentation interface.

For a description of the valid syntax for the various parameters see the subsection titled "Parameter Syntaxes" later in this chapter.

Some of these parameters have default values which you will not be prompted for. To override these defaults, turn on prompting for the appropriate class of parameter. See the subsection "Prompt Level" under "Utilities" for more information.

APLI Parameters

parameter	syntax	description
ap_my_psap	paddr	Specifies the local Presentation Address which we will either establish our connection through or receive a connection.
ap_dst_psap	paddr	Specifies the Presentation address which we will attempt to connect to. There should already be a process listening on this specified address.
ap_data_file	string	Specifies a file which contains hex data to be sent to the remote across a connection.
ap_data_val	hex	Specifies an octet string to be sent to the remote. For large strings it is better to use ap_data_file instead.
ap_cdata_file	string	Specifies a file which contains hex data to be sent over the connection request (or response).
ap_cdata_val	hex	Specifies an octet string to be sent over the connection request (or response) to the remote. For large strings it is better to use ap_cdata_file instead.
ap_rdata_file	string	Specifies a file which contains hex data to be sent on a release request to the remote.
ap_rdata_val	hex	Specifies an octet string to be sent over a release request to the remote. For large strings it is better to use ap_rdata_file instead.

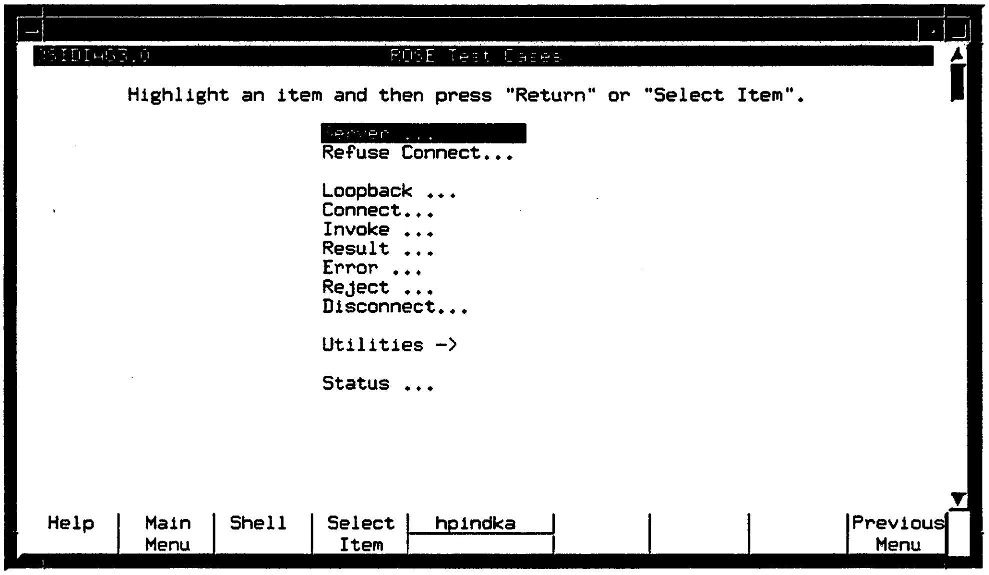
APLI Parameters (continued)		
parameter	**syntax**	**description**
ap_appl_ctx	objid	Specifies the application context to be used for this connection.
ap_p_ctx_list	pcdl	Specifies the proposed contexts for this connection.
ap_envelope	yes_no	Specifies whether user data will be encapsulated by osidiag with ASN.1 information.
ap_pci	integer	Specifies the presentation context identifier to be used if osidiag encapsulates using ASN.1.

ROSE Tests

Osidiag allows you to verify the ability to connect to and receive connections from a remote node through the ROSE Interface. This interface relies heavily on the APLI operations. Osidiag also provides the ability to issue and respond to invocation requests.

Test Menu

The test case menu displayed for the ROSE tests is shown below:



```
OSIDIAG3.0                         ROSE Test Cases

        Highlight an item and then press "Return" or "Select Item".

                         Server ...
                         Refuse Connect...

                         Loopback ...
                         Connect...
                         Invoke ...
                         Result ...
                         Error ...
                         Reject ...
                         Disconnect...

                         Utilities ->

                         Status ...
```

Server

The server test will listen for an inbound connection and accept it. After accepting the connection the test will listen for other ROSE indications such as invocation requests or disconnect. Indications other than the disconnect will be reported and conditionally osidiag will issue the ROSE result operation to respond to an invocation.

The test will wait for a default duration of 30 seconds. If after 30 seconds no inbound connection is received the test will terminate with a timeout failure. The timeout duration can be modified by going to the utilities menu and selecting "Wait Time" prior to the execution of the test. This timeout also holds for subsequent listens for data and disconnect indications.

This test can also be used to receive single events and then return control to the menu. See the subsection "Save Connections" under Utilities.

Refuse Connect

This test is identical to the APLI Refuse Connect test.

Connect

The connect test will result in a connection request being sent to the remote ROSE peer. The operation is identical to that performed via the APLI connection test, with the addition of a call to the ro_bind() operation. After establishing the connection we release the connection by sending a release.

This test can also be used to establish and leave open a connection. See the subsection "Save Connections" under Utilities.

By default the connection will go out only over the configured CLNS subnetwork. This determination is based on the network address used for the local Presentation Address (parameter ap_my_psap). You should run a loopback test over CONS, or change the prompt level under Utilities (to include Local SAP) in order to run this test over the non-default subnetwork.

Loopback

The loopback test will perform the functions of the server and connect tests in one test. A listen will be posted for an incoming connection and then we will issue a connection to ourselves. After the connection has been successfully established it will be torn down.

When running this test you will be prompted for the destination address. If you only have one link configured then the default will suffice. If you have both LAN and X.25 links configured the default will correspond to LAN. To test over X.25 in this case, change the network address to that configured for CONS under osiconf.

Invoke

This test will send a ROSE invoke request over a ROSE connection. If no connection currently exists, one will be established. The user is prompted for the information to be carried on the invocation. Osidiag will then listen for a response PDU from the remote.

To perform multiple operations with the remote, you must choose the "Save Connections" option under the Utilities menu. Otherwise the connection will be shut down after the single invoke is performed.

Result

This test allows you to manually respond to an invoke request received from the remote. Note that the server by default will automatically respond to received invocation requests. If no connection currently exists, one will be established. The user is prompted for the information to be carried on the response.

Typically this operation should only be run when the following are true:

- You are running in "Save Connections" mode.
- You have received an invocation by running the "Server" test.
- The server was not "auto-responding" to the invocation.

Error

This test allows you to manually report an error against an invoke request received from the remote. The same conditions as described for "Result" above should be met for this operation.

Reject

This test allows you to manually issue a reject request. The same conditions as described for "Result" above should be met for this operation.

Disconnect

This test is only useful when saving connections across tests (as described in the "Save Connections" subsection of the Utilities section). It will shut down the existing connection and the registration with the stack.

Utilities

By selecting this item you will be taken to the Utilities menu. To return from the Utilities to the ROSE tests press Exit.

Status

This operation invokes the program otsstat(1M) to display the current status of OTS. It also invokes osiconfchk(1M) to display a summary of important configured values (e.g. local addresses, destination systems, etc.).

Parameters

The following describes the parameters which osidiag will request during the various tests.

Remote APRI Presentation Address

When running any test which requires a connection to be established a window is brought up for you to enter the address of the remote entity which we will attempt to connect. The window is the same used for APLI.

Proposed Presentation Context Definition List

When establishing a connection you will be prompted for the a list of Presentation Contexts to be used for the life of the connection. The window is the same used for APLI.

APRI Application Context

When establlishing a connection you must also specify the application context. The window is the same used for APLI.

ROSE Invocation Data

When transfering data you will be prompted with a window which has the same fields as described for APLI.

The field "ASN.1 Envelope" indicates whether osidiag should enclose the data value specifed with ASN.1 headers, to guarantee that it is complies with the ASN.1 encoding rules.

The following ASN.1 structure is used for ROSE data:

```
    OCTETSTRING UserData;
```

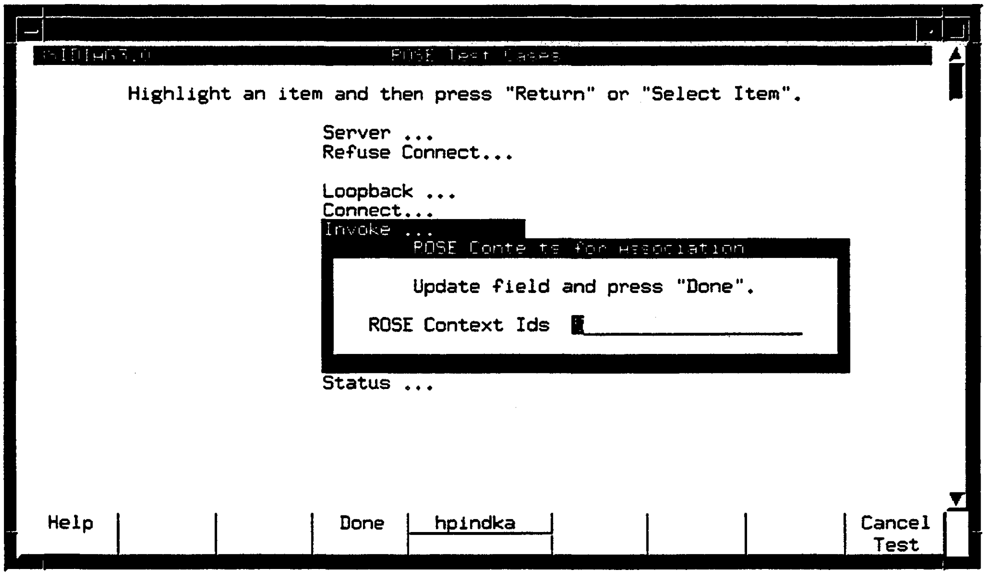
Note if the Envelope Flag is set to "Y" then the ASN.1 header will be sent carrying any user data present in the Data File or Value fields. If no data is specified an empty ASN.1 header will still be sent.

The PCI field specified for ROSE is passed as an argument to the ROSE function and is required whether or not the ASN.1 envelope flag is set.

ROSE Contexts for Association

When establishing connections through ROSE, osidiag must specify which contexts defined for the connection are available to ROSE. This argument is passed on the ro_bind() command.

The following window is presented.

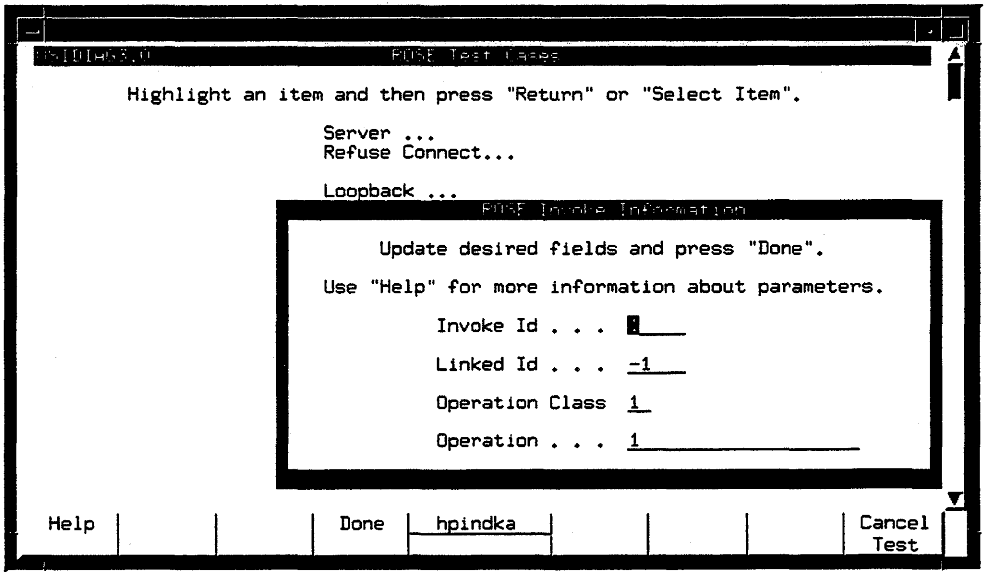


```
Highlight an item and then press "Return" or "Select Item".

                    Server ...
                    Refuse Connect...

                    Loopback ...
                    Connect...
                    Invoke ...
                          ROSE Contexts for Association

                          Update field and press "Done".

                      ROSE Context Ids  ________________

                    Status ...
```

You may enter one or more integer values separated by a space. Each value should be a valid PCI which was defined as part of the Presentation Context List (see APLI).

ROSE Invoke Information

When issuing an invocation request the following screen will be presented to specify information carried on the invocation. This information is in addition to the argument information passed on a subsequent screen.



```
OSIDIAG3.0                    ROSE Test Cases

     Highlight an item and then press "Return" or "Select Item".

                         Server ...
                         Refuse Connect...

                         Loopback ...
                            ROSE Invoke Information

                    Update desired fields and press "Done".

                 Use "Help" for more information about parameters.

                         Invoke Id . . . ____

                         Linked Id . . . -1__

                         Operation Class 1_

                         Operation . . . 1_______________

  Help  |      |      |  Done  | hpindka |      |      |  Cancel
                                                           Test
```

The Invoke Id is a value used to uniquely identify the invocation from any other outstanding invocations.

The Linked Id indicates if this invocation is being issued as a side effect of another invocation. A value of -1 indicates no linking.

The operation class is currently not used.

The operation code may be either an integer value, or an object id. It describes what operation is to be performed by the remote. Osidiag will display this value when it receives an invocation, but attaches no meaning to different values.

Example

The following shows a successful invoke test.

```
entity:    ROSE
test case: Invoke
------------------------------------
ap_my_psap          : "aprd"."aprd"."aprd"."hpindka"
ap_dst_psap         : "aprd"."aprd"."aprd"."hpindka"
ap_p_ctx_list       : {{3 {2 2 1 0 1}{{2 1 1}}}{5 {2 1 9999 1}{{2 1 1}}}}
ap_appl_ctx         : 2 1 9999 2
rose_pci_list       : 5
rose_invoke_id      : 1
rose_link_id        : -1
rose_op_class       : 1
rose_op             : 1
rose_arg_file       :
rose_arg_val        : "argument"
rose_pci            : 5
rose_envelope       : Y
------------------------------------
ap_open()              fd: 09   result: 0  ap_errno: 0x0
ap_init_env()          fd: 09   result: 0  ap_errno: 0x0
ap_snd(A_ASSOC_REQ)    fd: 09   result: 0  ap_errno: 0x0
ap_poll()              fd: 09   result: 0  ap_errno: 0x0
ap_rcv()               fd: 09   result: 0  ap_errno: 0x0
                         indication:  A_ASSOC_CNF
            result:  AP_ACCEPT
            data:  (HEX)                                       (ASCII)
            BE 80 28 05 02 01 03 81 00 00 00                   ..(........
ro_bind()              fd: 09   result: 0  ap_errno: 0x0
ap_snd(RO_INVOKE_REQ)   fd: 09   result: 0  ap_errno: 0x0
ap_poll()              fd: 09   result: 0  ap_errno: 0x0
ap_rcv()               fd: 09   result: 0  ap_errno: 0x0
                         indication:  RO_RESULT_IND
                         context id:      5
                         invoke id:       1
                         operation:       1
            data:  (HEX)                                       (ASCII)
            04 00                                              ..
ap_snd(A_RELEASE_REQ)   fd: 09   result: 0  ap_errno: 0x0
ap_poll()              fd: 09   result: 0  ap_errno: 0x0
ap_rcv()               fd: 09   result: 0  ap_errno: 0x0
                         indication:  A_RELEASE_CNF
            data:  (HEX)                                       (ASCII)
            BE 80 28 05 02 01 03 81 00 00 00                   ..(........
ro_unbind()            fd: 09   result: 0  ap_errno: 0x0
ap_close()             fd: 09   result: 0  ap_errno: 0x0
```

```
-----------------------------------
test status:  PASSED
======================================================================
```

The first two lines indicate the test being executed. The second section of output indicates the parameters used to execute the test. The third section of output indicates the dialog which took place with the remote. Finally the last section indicates the test status.

Parameter Display

The meaning of the parameters is given in the table at the end of this section. Many of the parameters beginning with "ap_" are defined by APLI. See also the APLI parameter summary.

Dialog Display

As the ROSE test runs it displays its interaction with the interface and its remote peer. Many of the operations for establishing and releasing the connection are from APLI. See the APLI example for details about these. A description of the ROSE operations performed is given below:

- **ro_bind()** - Initializes this connection to use ROSE, and associates the PCI's set in the rose_pci argument with ROSE. No remote dialog results from this call.
- **ap_snd(RO_INVOKE_REQ)** - All ROSE primitives, like APLI, are sent via the same function, ap_snd(). An argument to this function indicates what primitive is to be sent. In this case osidiag is sending an Invoke Request.
- **ap_rcv()** - this operation is used to receive all APLI indications and confirmations. Upon successful completion, the indication code is displayed.

  For ROSE operations, the important values associated with the primitive indication are displayed, such as the invocation id, PCI, etc.

  Last any data carried on that indication is shown. Note that the data will typically be ASN.1 encoded, as in the example above.
- **ro_unbind()** - release the association between ROSE and the APLI endpoint.

Test Status

Lastly the status of the test is displayed. In this case the test passed. See the next section for information on failed tests.

Interpreting Errors

If a ROSE test operation fails the same information as displayed for APLI will be given. See the APLI section for discussion.

ROSE Parameters

The following table describes the parameters used for the ROSE interface.

Many parameters are shared between APLI and ROSE, for instance local and remote address. See the APLI section for discussion of these parameters.

For a description of the valid syntax for the various parameters see the subsection titled "Parameter Syntaxes" later in this chapter.

Some of these parameters have default values which you will not be prompted for. To override these defaults, turn on prompting for the appropriate class of parameter. See the subsection "Prompt Level" under "Utilities" for more information.

ROSE Parameters		
parameter	**syntax**	**description**
rose_auto_res	yes_no	Specifies whether invocations should be automatically responded to when in server mode.
rose_envelope	yes_no	Specifies whether invoke or result data will be encapsulated by osidiag with ASN.1 information.
rose_pci	integer	Specifies the presentation context identifier to be used for ROSE operations.
rose_pci_list	intlist	Specifies the presentation context identifier(s) which can be used for carrying ROSE information.
rose_invoke_id	integer	Specifies the invocation id for various ROSE operations.
rose_link_id	integer	Specifies the linked ROSE invocation id for various ROSE operations.
rose_op_class	integer	This parameter is exposed through ROSE but has no effect.
rose_op	refid	Specifies the error value to be carried on a RO_ERROR.
rose_err_val	refid	Specifies the error value to be carried on a RO_ERROR.
rose_prob_type	integer	Specifies the problem type to be carried on a RO_REJECT.
rose_rej_reason	integer	Specifies the reject reason to be carried on a RO_REJECT.

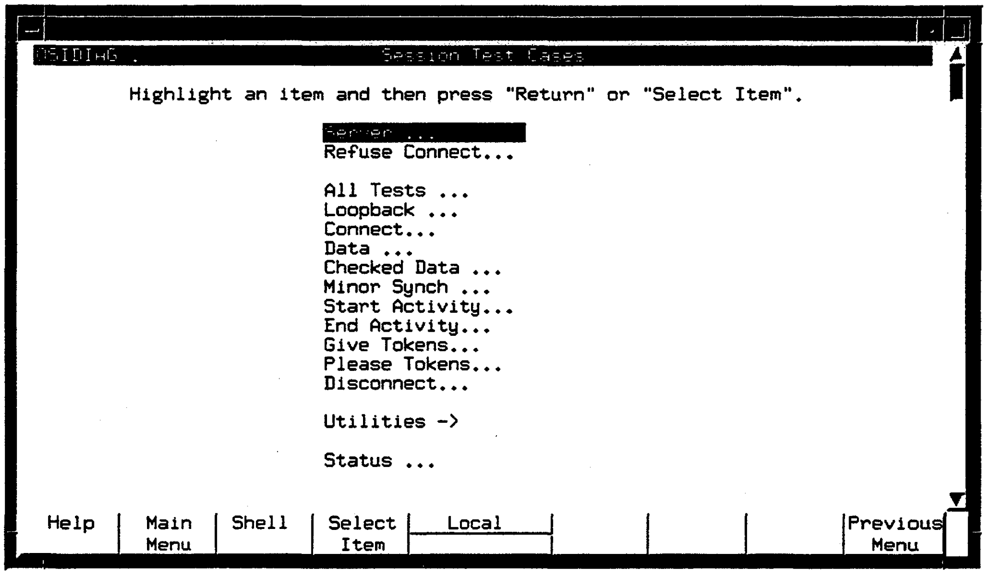
ROSE Parameters (continued)		
parameter	syntax	description
rose_arg_file	string	Specifies a file which contains hex data to be sent to the remote across along with an invocation.
rose_arg_val	hex	Specifies an octet string to be sent to the remote. For large strings it is better to use rose_arg_file instead.
rose_res_file	string	Specifies a file which contains hex data to be sent over a ROSE result request.
roes_res_val	hex	Specifies an octet string to be sent on the ROSE result request. For large strings it is better to use rose_res_file instead.

Session Tests

Osidiag allows you to verify that the Session layer of OTS can both connect and receive connections from the remote. In addition several other Session layer primitives may be issued across a connection.

Test Menu

The test case menu displayed for the Session tests is shown below:



```
OSIDIAG .                       Session Test Cases

        Highlight an item and then press "Return" or "Select Item".

                           Server ...
                           Refuse Connect...

                           All Tests ...
                           Loopback ...
                           Connect...
                           Data ...
                           Checked Data ...
                           Minor Synch ...
                           Start Activity...
                           End Activity...
                           Give Tokens...
                           Please Tokens...
                           Disconnect...

                           Utilities ->

                           Status ...

  Help  | Main | Shell | Select |  Local  |  |  |  |Previous
        | Menu |       | Item   |         |  |  |  | Menu
```

Server

The server test will listen for an inbound connection and accept it. After accepting the inbound connection the test will listen for other indications such as data, or release. The type of indication received will be displayed. If a release is received the test will respond to the request and return to the test menu. If a data indication is received the contents will be displayed and another indication will be listened for. For other primitives such as give tokens and activity start the test will respond appropriately. If the primitive is unsupported, osidiag will abort the connection.

The test will wait for a default duration of 30 seconds. If after 30 seconds no inbound connection is received the test will terminate with a timeout failure. The timeout duration can be modified by going to the utilities menu and selecting "Wait Time" prior to the execution of the test.

This test can also be used to receive single events and then return control to the menu. See the subsection "Save Connections" under the Utilities section.

Refuse Connect

This test is similar to the Server test in that it will wait for an incoming connection indication. The difference is that rather than accepting the connection it will return an refuse.

All Tests

This test will run through the connection test as well as attempt to issue all the primitives supported through osidiag tests. You need to have an osidiag Session Server running to respond to the requests issued.

Loopback

This test combines the Server and Connect tests into a single test operation. It will post a listen and then issue a connect to itself. The connection will then be dissolved. This is the simplest Session layer test for verifying the operation of the local Session layer.

When running this test you will be prompted for the destination address. If you have both LAN and X.25 configured then you will need to change the default Network Address in order to test X.25. Otherwise the default value will suffice.

Connect

The connect test will result in a connection request being sent to the remote Session peer. After a confirmation is received from the remote the connection is dissolved.

This test can also be used to establish and leave open a connection. See the subsection "Save Connections" under Utilities.

Data

This test will send a data PDU over a Session connection. If no connection currently exists, one will be established.

Checked Data

Checked data involves sending a specific data pattern to the osidiag Session Server. The server will verify that the data upon receipt still has the correct contents and send an acknowledgement back.

Because this test expects a special acknowledgement to be returned indicating that the data is correct it will not work against non-HP implementations.

Note changes to the C.03.00 and later versions of osidiag, may prevent it from performing the Checked Data test successfully with C.02.00 and earlier versions of osidiag. This is an incompatability just in this test in osidiag and should not be taken as an indication that the two versions do not interoperate.

Minor Synch

This test sends minor synchronization PDU to the remote. This packet can only be sent while we are within an activity. This implies that the Start Activity test must be performed first over the same connection. See the discussion of "Save Connections" in the "Utilitites" section.

Start Activity

This test sends the Session start activity PDU to the remote. This would typically be used while in "Save Connection" mode, see the "Utilities" section.

End Activity

This test sends the Session end activity PDU. The sender should have previously started an activity. This should be run in "Save Connections" mode as described in the "Utilities" section.

Give Tokens

This operation sends a give tokens PDU to the remote. This operation is dependent on our being in an activity.

Please Tokens

This operation sends a please tokens request to the remote. The HP Session server will generate a Give Tokens request when it receives a please token.

Disconnect

This test is only useful when saving connections across tests (as described in the "Save Connections" subsection of the Utilities section). It will shut down the existing connection and the registration with the stack.

Utilities

By selecting this item you will be taken to the Utilities menu. To return from the Utilities to the Session tests press Exit.

Status

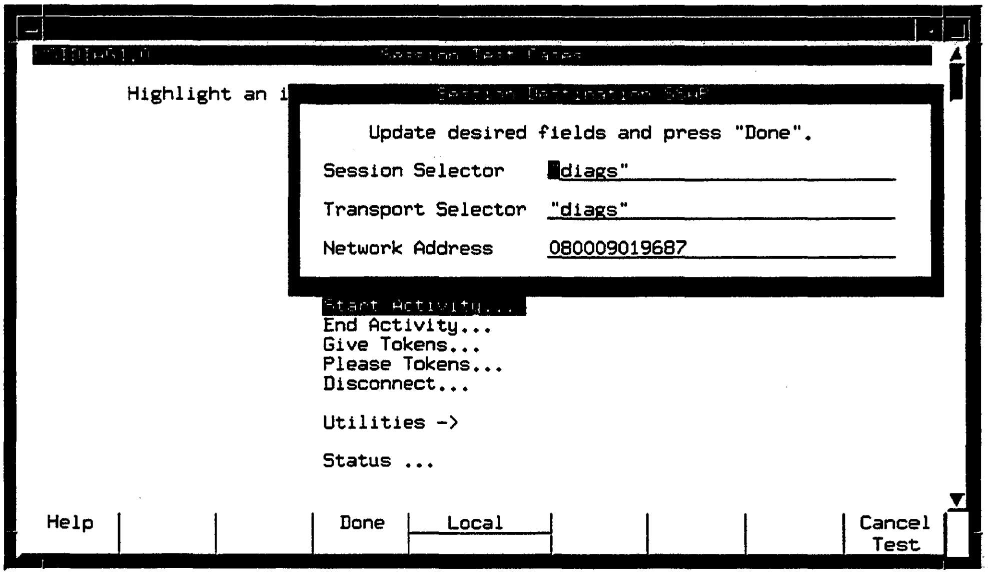
This operation runs "otsstat" to display whether the OTS stack is up and running.

Parameters

The following sections explain the parameters which you will be prompted for when running the various Session tests.

Destination Address

The data entry window presented by osidiag in order to get the address of the remote is shown below:



```
Highlight an i
                Update desired fields and press "Done".
            Session Selector     "diags"
            Transport Selector   "diags"
            Network Address      080009019687

            Start Activity...
            End Activity...
            Give Tokens...
            Please Tokens...
            Disconnect...

            Utilities ->

            Status ...
```

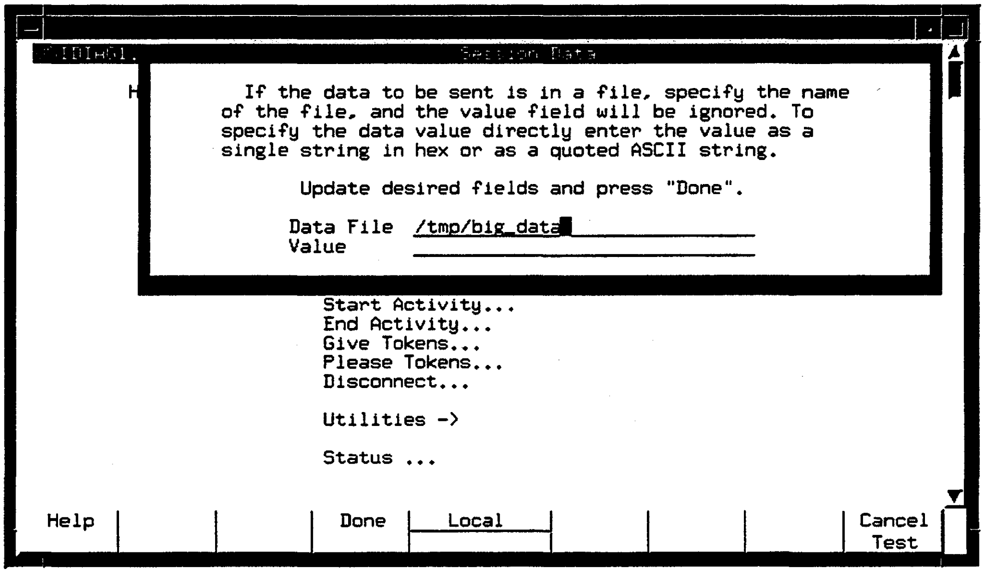
The Session address of the remote is composed of three components: the **session selector**, the **transport selector** and the **network address**. These fields should contain the Session address which is configured for the remote session peer.

The default value presented by osidiag is "diags"."diags".<local net addr>. Where local net addr is the local network address configured for this node. The hex value for "diags" is 6469616773. When running the Loopback test or the Connect and Server on the same node this will be sufficient.

Note that if the selector values are all ASCII that you may enter them as quoted strings. Otherwise specify them in hex.

Session Data PDU Contents

The following form is displayed when you run the data transfer test. A similar screen is displayed for other operations which allow data to be transfered.



```
  If the data to be sent is in a file, specify the name
of the file, and the value field will be ignored. To
specify the data value directly enter the value as a
single string in hex or as a quoted ASCII string.

       Update desired fields and press "Done".

      Data File  /tmp/big_data
      Value
```

Data File

Entering the name of a file in this field directs osidiag to read the data to be sent from that file, ignoring any information in the Value field. If you are sending a very large amount of data, or you are sending the same data across several tests, then it may be worthwhile to store the data in a file.

Data stored in hex files should be byte values separated by spaces. An example hex file is shown below:

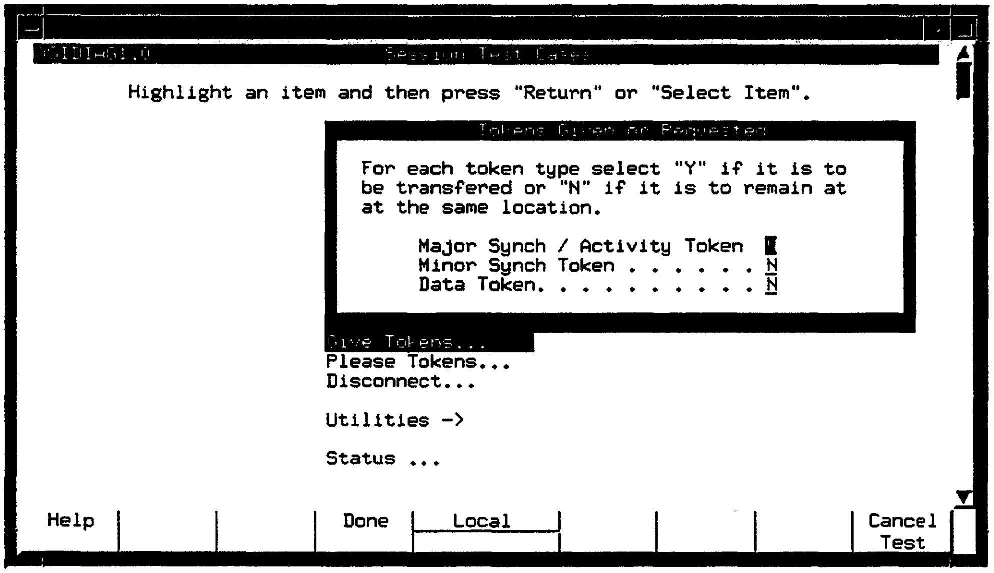
```
#
# Sample data file for Session test
#
A1 06 02 01 12
02 01 05
```

Value

This field is used to specify the data if no file name is given. The data value is specified in hex or quoted ASCII.

Tokens

The following form is displayed when you run a test which allows you to give or request tokens.



```
OSIDIAG1.0                    Session Test Cases

      Highlight an item and then press "Return" or "Select Item".

                          Tokens Given or Requested

                     For each token type select "Y" if it is to
                     be transfered or "N" if it is to remain at
                     at the same location.

                          Major Synch / Activity Token _
                          Minor Synch Token . . . . . . N
                          Data Token. . . . . . . . . . N

                    Give Tokens...
                    Please Tokens...
                    Disconnect...

                    Utilities ->

                    Status ...
```

Answering "Y" next to a token type indicates that that token will be given or requested, depending on the PDU type.

Example

The following shows a successful connection test.

```
entity:    SESSION
test case: Connection
------------------------------------
ss_my_mbox          : /etc/net/osi/ots/mailbox/MBXDIAGI
ss_my_sap           : 0
ss_my_ssap          : ."diags"."diags"
ss_dst_ssap         : "diags"."diags"."hpindka"
ss_full_dup         : N
ss_user_ref         :
ss_com_ref          :
ss_addl_ref         :
ss_cdata_val        :
ss_cdata_file       :
ss_rdata_val        :
ss_rdata_file       :
------------------------------------
osi_init()                                  result: 0
osi_rgr_rq()          reg_id:   0x00000001   result: 0
osi_rgr_cf()          reg_id:   0x00000001   result: 0
ses_connect_rq()      conn_id:  0x00000001   result: 0
osi_get_event()       conn_id:  0x00000001   result: 0
                        indication:  ESCONCF  (S_Connect.cnf)
ses_connect_cf()      conn_id:  0x00000001   result: 0
ses_release_rq()      conn_id:  0x00000001   result: 0
osi_get_event()       conn_id:  0x00000001   result: 0
                        indication:  ESRELCF  (S_Release.cnf)
ses_release_cf()      conn_id:  0x00000001   result: 0
osi_exit()            reg_id:   0x00000001   result: 0
------------------------------------
functional units:            0x8249
     HALF DUPLEX
     MINOR SYNCHRONIZATION
     ACTIVITY MANAGEMENT
     EXCEPTIONS
version number:              2
initial serial number:       -1
initial token position:      0x0
connection id:
  user reference:            NOT PRESENT
  common reference:          NOT PRESENT
  additional reference:      NOT PRESENT
------------------------------------
test status:  PASSED
==========================================================================
```

The first two lines indicate the test being executed. The second section of output indicates the parameters used to execute the test. The third section of output indicates the dialog which took place with the remote. The fourth section shows the contents of the connection accept received. Finally the last section indicates the test status.

Parameter Display

The meaning of the parameters is given in the table at the end of this section.

Dialog Display

As the Session test runs it displays its interaction with the stack and its remote peer. A description of each operation performed is given below:

- **osi_init()** - initializes the Session library. **osi_rgr_rq()** - registers this application with the local stack. No communication with the remote results from this call.
- **osi_rgr_cf()** - retrieves the confirmation from the local stack regarding our registration.
- **ses_connect_rq()** - sends a Session layer connection request to the remote node.
- **osi_get_event()** - listens for messages from the remote node. By default we will wait a maximum of 30 seconds for a message before timing out (see the Utilities section entry for "Wait Time" to adjust this value).

  If a message arrives its type is displayed on the next line of output. In this case we are expecting a confirmation for the connection request we sent. The first item "ESCONCF" is an internal name for the indication and the second item, "S_Connect.cnf", is just a more readable form.
- **ses_connect_cf()** - this operation decodes at the Session level the connection confirmation. Any data carried on the Session level packet is displayed. In this case no data was carried.
- **ses_release_rq()** - sends a request to the remote to release the connection.
- **osi_get_event()** - listens for messages from the remote node. In this case we are expecting a confirmation for the release request we sent.
- **ses_release_cf()** - decodes at the Session level the release confirmation received. Any data carried on this confirmation is displayed, in this case none.
- **osi_exit()** - release all resources used by the Session library, including any registrations.

Negotiated Parameters

The functional units which were negotiated are displayed. The hex value shown is the value of the functional units bitstring. The indented items appearing below it are the names of the units corresponding to the bits which are enabled.

The initial serial number indicates the serial numbers which will be used to track activity management commands. A value of -1 indicates no serial number is given.

The initial token position is a one byte value indicating which side of the connection various tokens reside. To meaning of the value is as follows:

```
+---------------+
|t t M M m m d d|
+---------------+

 tt = Negotiated Release token
 MM = Major Synch / Activity token
 mm = Minor Synch token
 dd = data token
```

For each two bit field a value of 0 indicates the token is at the initiator, a value of 1 indicates at the acceptor, 2 indicates it is the acceptor's choice and 3 indicates it is the provider's choice.

Test Status

Lastly the status of the test is displayed. In this case the test passed. The next section describes what is displayed if the test fails.

Interpreting Errors

If a Session test operation fails, the following output will appear in the status section at the end of the output:

```
------------------------------------
test status:  FAILED
operation:    ses_pab_id()
return code:  -89


osidiag:  Received service provider abort.
     Check the reason code with the abort, and check the remote status.
     This may indicate a bad address, or a bad set of negotiated parameters.
     It can also indicate failure at the lower layers of the stack or
     unexpected termination of the remote application.
=========================================================================
```

The meaning of each field is as follows:

- **test status** - indicates whether we passed or failed. Any error encountered during the operation (including setting up tracing and logging, if requested) will result in a status of failed.
- **operation** - indicates at what point in the test we encountered the first error. Typically this will be the name of a Session interface operation.
- **return code** - this is the return value from the Session interface call.
- **Formatted error text** - after the above fields are displayed, osidiag will provide any textual information about the error. Typically this is displaying the meaning of the return code. In this case "osidiag: Received service provider abort."

Further Information

Further information on interpreting errors may be found in the Troubleshooting section of this manual. Also check the Session Programmer's manual.

Parameter Summary

The following table describes the parameters used for Session.

For a description of the valid syntax for the various parameters see the section titled "Parameter Syntaxes".

Note that some parameters have defaults values which you are not prompted for. To override these defaults, turn on prompting for the appropriate class of parameter. See the subsection "Prompt Level" under "Utilities" for more information.

Session Parameters		
parameter	**syntax**	**description**
ss_my_ssap	local address	Specifies the Session and Transport selectors which we will either establish our connection through or receive a connection.
ss_dst_ssap	ssap	Specifies the Session address which we will attempt to connect to. There should already be a process listening on this specified address.
ss_my_mbox	string	Specifies the name of the pipe file which we will use for communicating with the OTS stack process.
ss_data_file	string	Specifies a file which contains hex data to be sent to the remote across a connection.
ss_data_val	hex	Specifies an octet string to be sent to the remote. For large strings it is better to use ss_data_file instead.
ss_cdata_file	string	Specifies a file which contains hex data to be sent over the connection request (or response).
ss_cdata_val	hex	Specifies an octet string to be sent over the connection request (or response) to the remote. For large strings it is better to use tp_cdata_file instead.
ss_rdata_val	hex	Specifies an octet string to be sent as user data on the release request (or response). For large strings it is better to use ss_rdata_file instead.

Session Parameters (continued)		
parameter	syntax	description
ss_udata_file	string	Specifies a file which contains hex data to be sent as user data on other operations (i.e. Activity Start, or Minor Synch).
ss_udata_val	hex	Specifies an octet string to be sent as user data on other operatoins (i.e. Activity Start, or Minor Synch). For large strings it is better to use ss_udata_file instead.
ss_rdata_file	string	Specifies a file which contains hex data to be sent as user data on the release request (or response).
ss_user_ref	octet string	This gives any data to be carried as the User Reference field on a Session connect request.
ss_com_ref	octet string	This gives any data to be carried as the Common Reference field on a Session connect request.
ss_addl_ref	octet string	This gives any data to be carried as the Additional Reference field on a Session connect request.
ss_full_dup	yes_no	This flag indicates if osidiag will negotiate Full or Half duplex for the connection. By default Half duplex is requested (the flag is "N").
ss_act_id	octet string	This gives the activity id which will be sent on an activity start request.
ss_minor_token	yes_no	This flag indicates if we will pass the minor synch token on a give token request (also on a please token).
ss_data_token	yes_no	This flag indicates if we will pass the data token on a give token request (also on a please token).
ss_major_token	yes_no	This flag indicates if we will pass the major synch (activity start) token on a give token request (also on a please token).
ss_n_pkts	integer	Specifies the number of packets to be sent in a checked data test.
ss_pkt_size	integer	Specifies the size of each data packet to be sent in a checked data test.

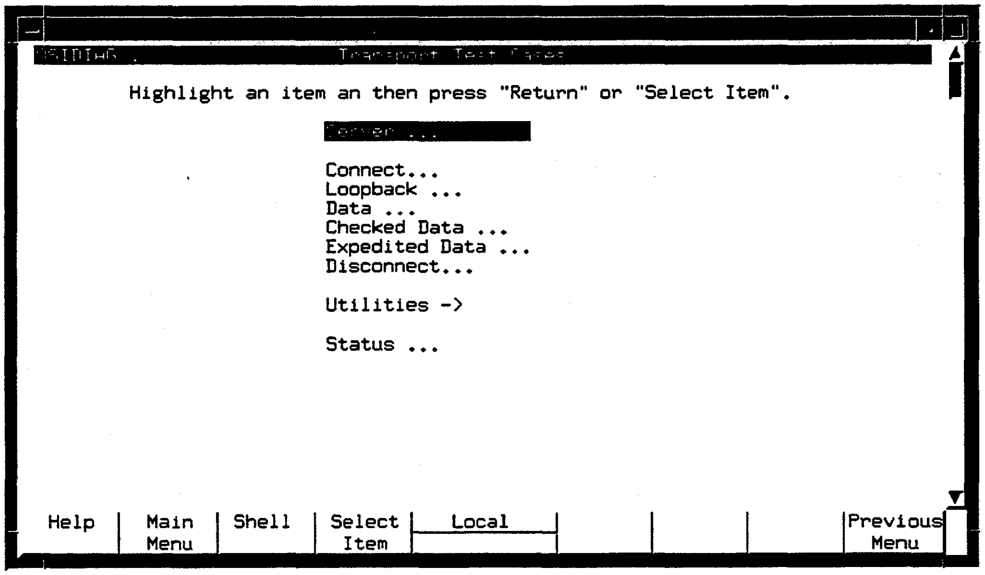
Transport Tests

Osidiag allows you to verify the ability to connect to and receive connections from a remote node at the Transport level. Also provided is the ability to send data and expedited data to the remote over a transport connection.

These tests make use of the X/Open standard XTI interface.

Test Menu

The test case menu displayed for the Transport tests is shown below:



```
OSIDIAG .                 Transport Test Cases

      Highlight an item an then press "Return" or "Select Item".

                        Server ...

                        Connect...
                        Loopback ...
                        Data ...
                        Checked Data ...
                        Expedited Data ...
                        Disconnect...

                        Utilities ->

                        Status ...
```

Server

The server test will listen for an inbound connection and accept it. After accepting the connection the test will listen for other Transport indications such as data, expedited data or disconnect. Indications other than the disconnect will be reported and osidiag will continue to listen. When a disconnect is received the test ends.

The test will wait for a default duration of 30 seconds. If after 30 seconds no inbound connection is received the test will terminate with a timeout failure. The timeout duration can be modified by going to the utilities menu and selecting "Wait Time" prior to the execution of the test. This timeout also holds for subsequent listens for data and disconnect indications.

This test can also be used to receive single events and then return control to the menu. See the subsection "Save Connections" under Utilities.

Connect

The connect test will result in a connection request being sent to the remote Transport peer. After the connection is confirmed it is dissolved by sending a disconnect.

This test can also be used to establish and leave open a connection. See the subsection "Save Connections" under Utilities.

By default the connection will go out only over the configured CLNS subnetwork. This determination is based on the network address used for our local Transport Address (parameter tp_my_tsap). You should run a loopback test over CONS, or change the prompt level under Utilities (to include Local SAP) in order to run this test over the non-default subnetwork.

Loopback

The loopback test will perform the functions of the server and connect tests in one test. A listen will be posted for an incoming connection and then we will issue a connection to ourselves. After the connection has been successfully established it will be torn down.

When running this test you will be prompted for the destination address. If you only have one link configured then the default will suffice. If you have both LAN and X.25 links configured the default will correspond to LAN. To test over X.25 in this case, change the network address to that configured for CONS under osiconf.

Data

This test will send a data PDU over a Transport connection. If no connection currently exists, one will be established. The user is prompted for the data to be carried in the PDU.

Checked Data

Checked data involves sending a specific data pattern to the osidiag Transport Server. The server will verify that the data upon receipt still has the correct contents and send an acknowledgement back.

Because this test expects an acknowledgement to be returned indicating that the data is correct it will not work against non-HP implementations.

Expedited Data

This test will send an expedited data PDU over a Transport connection. If no connection currently exists, one will be established. The user is prompted for the data to be carried in the PDU. HP's XTI allows a maximum of 16 bytes of expedited data be sent.

Disconnect

This test is only useful when saving connections across tests (as described in the "Save Connections" subsection of the Utilities section). It will shut down the existing connection and the registration with the stack.

Utilities

By selecting this item you will be taken to the Utilities menu. To return from the Utilities to the Transport tests press Exit.

Status

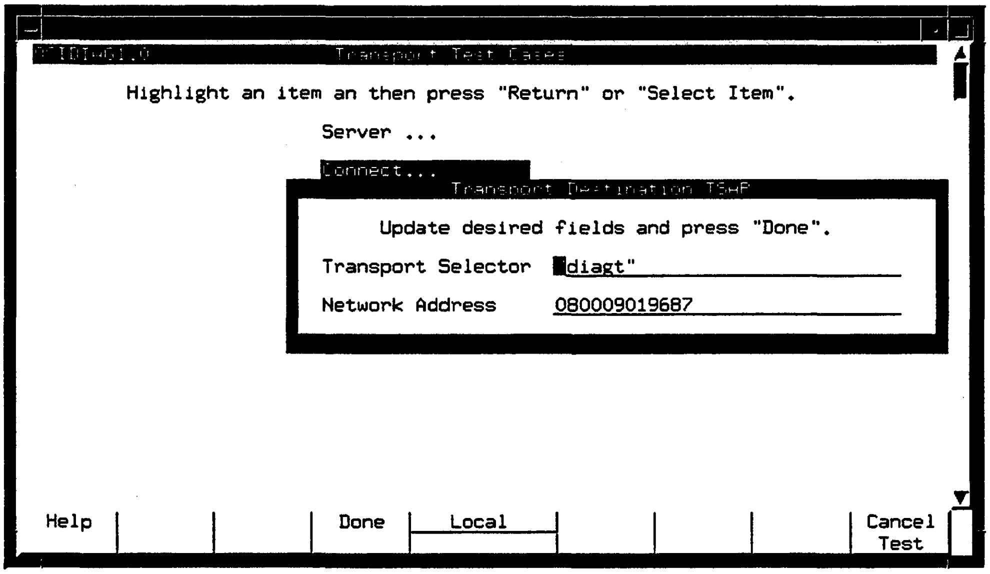
This operation invokes the program otsstat(1M) to display the current status of OTS. It also invokes osiconfchk(1M) to display a summary of important configured values (e.g. local addresses, destination systems, etc.).

Parameters

The following describes the parameters which osidiag will request during the various tests.

Destination TSAP

The following window is brought up for you to enter the address of the remote Transport entity which we will attempt to connect to.



```
OTIDIwG1.0              Transport Test Cases

     Highlight an item an then press "Return" or "Select Item".

                    Server ...
                    Connect...
                           Transport Destination TSAP

                      Update desired fields and press "Done".

                    Transport Selector   "diagt"
                    Network Address      080009019687
```

The Transport address of the remote is composed of two components: the **transport selector** and the **network address**. These fields should contain the Transport address which is configured for the remote session peer.

The default value presented by osidiag is "diagt".<local net addr>. Where local net addr is the configured local network address. The hex value for "diagt" is 6469616774. When running the Loopback test or the Connect and Server on the same node this will be sufficient.

When running against a remote HP node also running osidiag, you need only change the Network Address. For non-HP nodes this value should be obtained by asking the network administrator for the remote node.

Note that if the selector values are all ASCII that you may enter them as quoted strings. Otherwise specify them in hex.

You can determine the network address of a remote HP node by running osiconf on that node and viewing the subnetwork configuration.

Transport Data PDU Contents

When you run the data transfer test or expedited data tests you will be asked to supply the data to be carried. The form and its behavior is similar to the description for Data PDUs in the Session section of this chapter.

Example

The following shows a successful connection test.

```
entity:    TRANSPORT
test case: Connection
------------------------------------
tp_my_tsap          : .."diagt"
tp_dst_tsap         : "diagt"."hpindon"
tp_class            : 4
tp_alt_class        : N
tp_cdata_val        :
tp_cdata_file       :
tp_ddata_val        :
tp_ddata_file       :
------------------------------------
t_open()              fd: 10   result: 0  t_errno: 0
t_bind()              fd: 10   result: 0  t_errno: 0
t_connect()           fd: 10   result: 0  t_errno: 0
t_look()              fd: 10   result: 0  t_errno: 0
                        indication:  T_CONNECT
t_rcvconnect()        fd: 10   result: 0  t_errno: 0
t_snddis()            fd: 10   result: 0  t_errno: 0
t_unbind()            fd: 10   result: 0  t_errno: 0
t_close()             fd: 10   result: 0  t_errno: 0
------------------------------------
class:          4
flowctrl:       -1
------------------------------------
test status:  PASSED
=========================================================================
```

The first two lines indicate the test being executed. The second section of output indicates the parameters used to execute the test. The third section of output indicates the dialog which took place with the remote. The fourth section shows the contents of the connection accept received. Finally the last section indicates the test status.

Parameter Display

The meaning of the parameters is given in the table at the end of this section.

Dialog Display

As the Transport test runs it displays its interaction with the stack and its remote peer. A description of each operation performed is given below:

- **t_open()** - creates a transport endpoint (a file descriptor) to be used for our XTI communication. No remote dialog results from this call.
- **t_bind()** - associates our local Transport address with the endpoint.
- **t_connect()** - sends a Transport layer connection request to the remote node. Any user data pointed to by tp_cdata_val or tp_cdata_file will be carried on this PDU.
- **t_look()** - listens for messages from the remote node. By default we will wait a maximum of 30 seconds for a message before timing out (see the Utilities section entry for "Wait Time" to adjust this value).

  If a message arrives its type is displayed on the next line of output. In this case we received a confirmation for the connection request we sent. The name for this indication in <xti.h> is "T_CONNECT".
- **t_rcvconnect()** - this operation decodes at the Transport level the connection confirmation. Any data carried on the Transport level packet is displayed. In this case no data was carried.
- **t_snddis()** - sends a request to the remote to disconnect. No confirmation is passed up to the user because it carries no data and must confirm the disconnect.
- **t_unbind()** - disassociate ourselves from the transport endpoint.
- **t_close()** - close the file descriptor.

Negotiated Parameters

The class of service agreed upon between the local and remote transport entities is displayed.

Also the use of flow control is shown. This parameter is only meaningful for Class 2. A value of -1 indicates it is has no meaning. A value of 1 means flow control is used, 0 means it is not used.

Test Status

Lastly the status of the test is displayed. In this case the test passed. See the next section for information on failed tests.

Interpreting Errors

If a Transport test operation fails, the following output will appear in the status section at the end of the output:

```
------------------------------------
test status:   FAILED
operation:     t_rcvdis()
t_errno:       -83
errno:         0

osidiag:  Received unexpected disconnect.
     Examine the reported reason code for more information.  This may
     indicate that the address specified was incorrect.  Check the status
     of the remote entity for further information.
=======================================================================
```

The meaning of each field is as follows:

- **test status** - indicates whether we passed or failed. Any error encountered during the operation (including setting up tracing and logging, if requested) will result in a status of failed.
- **operation** - indicates at what point in the test we encountered the first error. Typically this will be the name of an XTI operation, since Transport is accessed via the XTI interface.
- **t_errno** - this is the value of the XTI errno returned by the failed XTI operation.

  In some instances (as the one shown above) this value will contain an osidiag specfic error. This happens when an operation completes without error, but the operation is not part of the correct order of the test. Here we successfully received a disconnect indication, but we were not expecting one.
- **errno** - this contains any system errno encountered during this operation. Some XTI errors may fill in both. A value of zero indicates no system error was returned.
- **Formatted error text** - after the above fields are displayed, osidiag will provide any textual information about the error. Typically this is displaying the meaning of the t_errno. In this case "osidiag: Received unexpected disconnect."

Further Information

Further information on interpreting errors may be found in the Troubleshooting chapter of this manual. Also check the XTI Programmer's manual.

Transport Parameters

The following table describes the parameters used for Transport.

For a description of the valid syntax for the various parameters see the subsection titled "Parameter Syntaxes" later in this chapter.

Some of these parameters have default values which you will not be prompted for. To override these defaults, turn on prompting for the appropriate class of parameter. See the subsection "Prompt Level" under "Utilities" for more information.

Transport Parameters		
parameter	**syntax**	**description**
tp_my_tsap	local addr	Specifies the local T-selector which we will either establish our connection through or receive a connection.
tp_dst_tsap	tsap	Specifies the Transport address which we will attempt to connect to. There should already be a process listening on this specified address.
tp_class	integer	Specifies the Transport class to be used on this connection.
tp_alt_class	yes_no	Specifies whether alternate class 0 should be specified on calls over class 2.
tp_data_file	string	Specifies a file which contains hex data to be sent to the remote across a connection.
tp_data_val	hex	Specifies an octet string to be sent to the remote. For large strings it is better to use tp_data_file instead.
tp_cdata_file	string	Specifies a file which contains hex data to be sent over the connection request (or response).
tp_cdata_val	hex	Specifies an octet string to be sent over the connection request (or response) to the remote. For large strings it is better to use tp_cdata_file instead.

Transport Parameters (continued)		
parameter	**syntax**	**description**
tp_xdata_file	string	Specifies a file which contains hex data to be sent as expedited data to the remote.
tp_xdata_val	hex	Specifies an octet string to be sent as expedited data to the remote. For large strings it is better to use tp_xdata_file instead.
tp_ddata_file	string	Specifies a file which contains hex data to be sent on a disconnect request to the remote.
tp_ddata_val	hex	Specifies an octet string to be sent over a disconnect request to the remote. For large strings it is better to use tp_ddata_file instead.
tp_n_pkts	integer	Specifies the number of packets to be sent in a checked data test.
tp_pkt_size	integer	Specifies the size of each data packet to be sent in a checked data test.

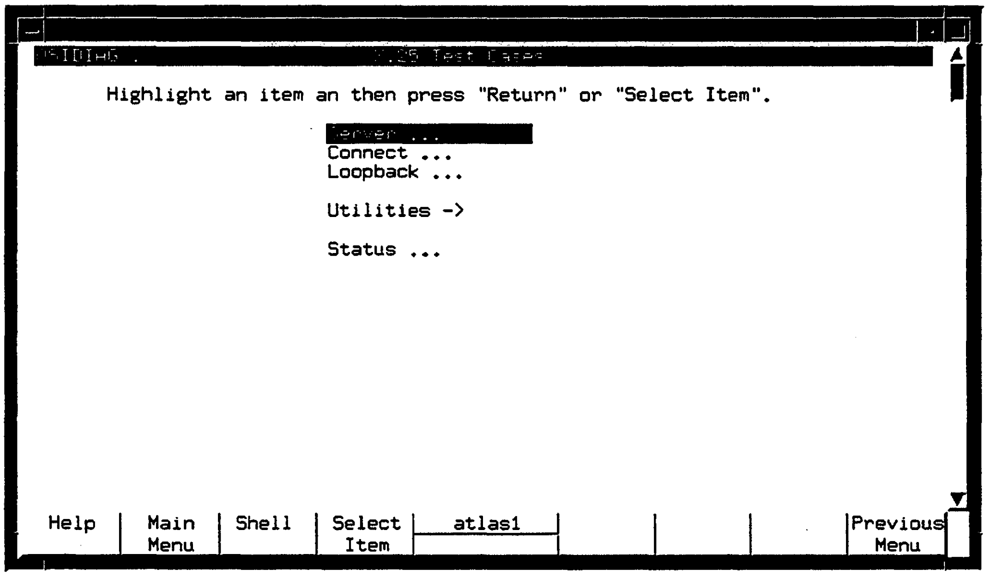
X.25 Tests

Osidiag allows you to verify the local node can both connect to and receive connections from the remote node through X.25.

Note the connect test differs from the CALL packets sent out by OTS in that osidiag never encodes ISO 8878 information, and osidiag never requests any special facilities (e.g. reverse charging).

Test Menu

The test case menu displayed for the X.25 tests is shown below:



```
OSIDIAG .                X.25 Test Cases

      Highlight an item an then press "Return" or "Select Item".

                          Server ...
                          Connect ...
                          Loopback ...

                          Utilities ->

                          Status ...
```

Server

The server test will post a listen for an inbound connection. When a connection arrives it will be accepted. The server will then wait for the first inbound frame and will display any data received to the screen. If the inbound frame is a clear request the diagnostic codes will be displayed.

This test will wait for a default of 30 seconds for an inbound connection. See the "Utilities" section on "Wait Timer" for information on adjusting this timeout.

Connect

The connect test will result in an X.25 CALL packet being sent to the remote node. If the call is accepted a clear is sent out on the line. Any errors encountered on the CALL are reported.

Loopback

This tests will post a listen for a connection and then send an X.25 CALL packet to itself. The connection is then CLEAR'ed. Any errors encountered are reported.

Utilities

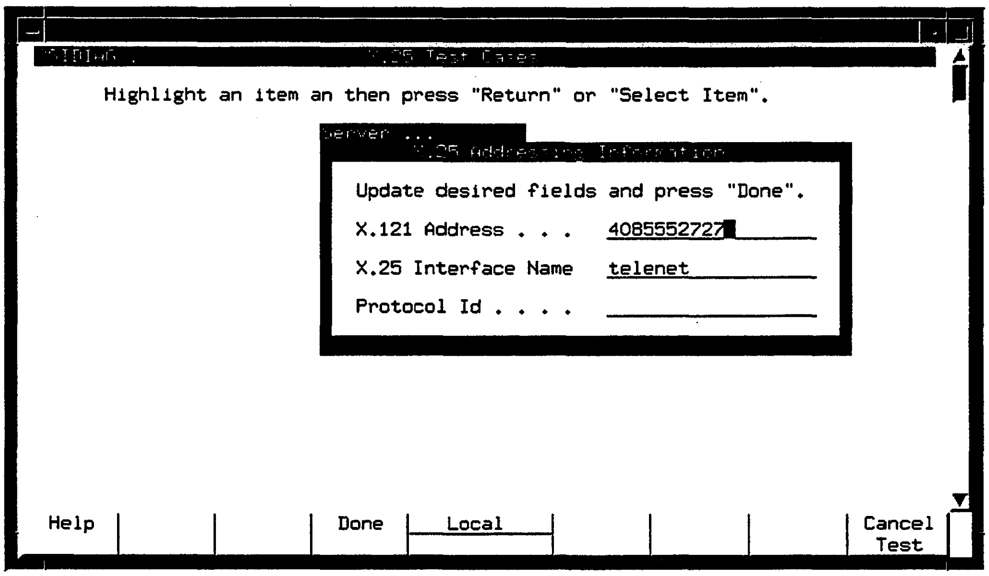
By selecting this item you will be taken to the Utilities menu. To return from the Utilities to the X.25 tests press Exit.

Status

This will invoke "x25stat" to display the status of the X.25 link and configuration information.

Addressing Information

The following popup is displayed for all the X.25 tests. In the case of the connect test it specifies the remote address to connect with. In the case of a server test it specifies the local address we will listen on. In the case of a loopback test it specifies both.



```
OSIDIAG :                 X.25 Test Cases

     Highlight an item an then press "Return" or "Select Item".

                    Server ...
                         X.25 Addressing Information

                    Update desired fields and press "Done".
                    X.121 Address . . .    4085552727
                    X.25 Interface Name    telenet
                    Protocol Id . . . .
```

X.121 Address

This value is the full X.121 Address including any subaddress if present. By default osidiag will set this value to the configured value for the first X.25 card which it finds configured in "/etc/net/osi/conf/ots_subnets". Any subaddress configured for this card will be ignored.

These addresses are specified in decimal.

X.121 Interface Name

This is the programmatic interface name for the card. Programs which access X.25 specify this name rather than a device file name. Osidiag will default this value to the first name configured for OTS.

Protocol Id

This value is used to further qualify incoming messages among X.25 users at the same X.121 address. See the discussion of X.25 in the "Addressing" chapter.

For osidiag this value is defaulted to the value "dx25". The protocol id used for CONS is 03010100 a blank value may also be used (for most stacks). The protocol id used for CLNS/X.25 is 81.

This value should be specified in hex or as a quoted ASCII string.

Example

The following shows a successful loopback test.

```
entity:    X25
test case: Loopback
------------------------------------
x25_addr             : 1111110007
x25_if_name          :
x25_pid              : "dx25"
------------------------------------
bind()                 errno: 0
listen()               errno: 0
connect()              errno: 0
accept()               errno: 0
close()                errno: 0
recv(*OOB*)            errno: 0
     05 00 00 F1 00                                .....
close()                errno: 0
------------------------------------
out of band event:  CLEAR
diagnostic code:    241
meaning:            Higher Level Initiated - Disconnect - normal
clear code:         0
meaning:            DTE Originated
------------------------------------
test status:  PASSED
=========================================================================
```

The first two lines indicate the test being executed. The second section of output indicates the parameters used to execute the test. The third section of output indicates the dialog which took place with the remote. The last section indicates the test status.

Parameter Display

The parameters displayed for the X.25 test are the same as those entered on the "Addressing Information" window. See above for description.

Dialog Display

As the X.25 test runs it displays its progress.

- **bind()** - this command associates the addressing information you specified with the our local X.25 connection endpoint.

- **listen()** - this command enables us to receive inbound X.25 connections on our endpoint.
- **connect()** - this command issues a connect request to the specified address.
- **accept()** - if this command completes it indicates we received a connection indication and have sent a response.
- **close()** - this command closes the initiating side of the connection. This will result in a disconnect being received.
- **recv(*OOB*)** - this indicates that we have received "Out of Band" data, such as a clear request. The data received is displayed on the following line. A decoded form of this is displayed in the Information Display section.
- **close()** - this closes our responding endpoint and ends the test.

Information Display

If any out of band data is received, a CLEAR or RESET indication, it will be decoded and displayed here. In this case we received a CLEAR request with a "Higher Level Initiated" cause code. This is normal.

Test Status

The status of the test is displayed. In this case the test passed. See the section on Common Errors for interpretation of failed tests.

Successful Connect Test

The information displayed for a connect test is a subset of what goes on for a loopback test. Specifically we should see just a **connect()** and a **close()** request being issued. This indicates that we successfully connected and then close the connection.

If a failure were to occur we would receive some "out of band" data with a diagnostic code describing the problem.

Successful Server Test

The information displayed for a connect test is a subset of what goes on for a loopback test. Specifically we should see a **bind()**, **listen()**, **accept()**, **recv(*OOB*)** and a **close()** request being issued. This indicates that we successfully received a connection. The out of band data should be the same as the loopback test.

Typical failure will be a timeout waiting for a connection.

Interpreting Errors

If an X.25 test operation fails, the following output will appear in the status section at the end of the output:

```
-----------------------------------
test status:  FAILED
operation:    socket()
return code:  221


EPROTONOSUPPORT:  Protocol not supported
======================================================================
```

The meaning of each field is as follows:

- **test status** - indicates whether we passed or failed. Any error encountered during the operation (including setting up tracing and logging, if requested) will result in a status of failed.
- **operation** - indicates at what point in the test we encountered the first error. Typically this will be the name of a socket operation, since the X.25 driver is accessed via sockets.
- **return code** - this is typically the value of the system errno returned on a socket system call. These are defined in the file "/usr/include/sys/errno.h" and in the manual page for errno(2).
- **Formatted error text** - after the above fields are displayed, osidiag will provide any textual information about the error. Typically this is displaying the meaning of the errno. In this case "EPROTONOSUPPORT: Protocol not supported", EPROTONOSUPPORT is the "#define" found in the errno.h file.

Out of Band Information

Many test failures will return indicating that a disconnect was received which indicated an error. To further analyze this problem you need to examine the Out of Band (OOB) information formatted before the status section.

```
-----------------------------------
out of band event:  CLEAR
diagnostic code:    67
meaning:            Call Setup Problem - Invalid Called Address
clear code:         13
meaning:            Not Obtainable
-----------------------------------
```

The above event tells us that our CALL request failed because of an "Invalid Called Address". More detailed information about these errors can be found by looking up the diagnostic code in the "Troubleshooting X.25/9000" manual.

Further Information

Further information on interpreting errors may be found in the Troubleshooting section of this manual. Also check the X.25 Troubleshooting manual.

X.25 Parameters

The following table describes the parameters used for X.25 tests.

For a description of the valid syntax for the various parameters see the section titled "Parameter Syntaxes".

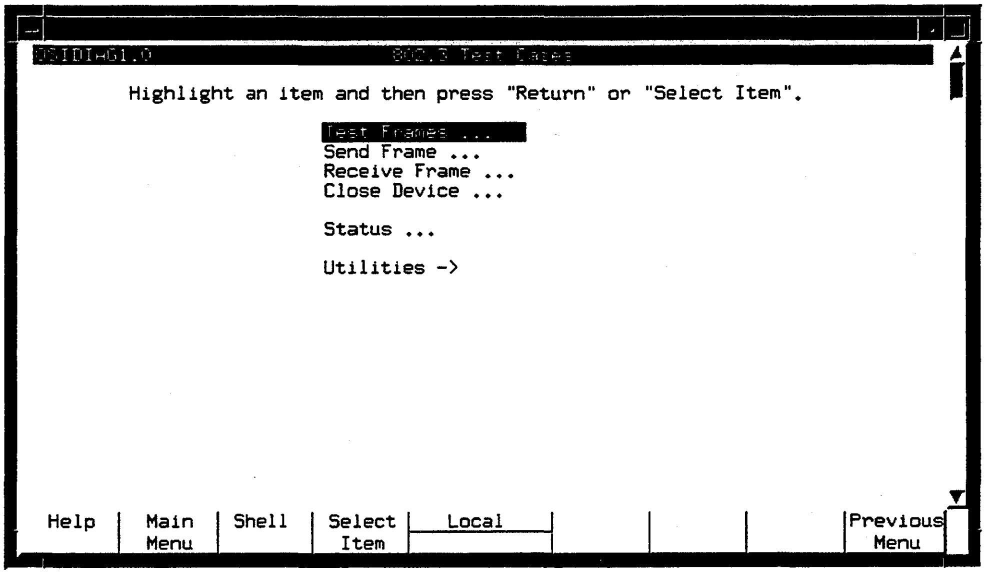
X.25 Parameters		
parameter	**syntax**	**description**
x25_addr	string	Specifies the remote X.121 address. This includes any subaddress which is to be used. Although the syntax is of type string only digits should appear.
x25_pid	octet string	Specifies the protocol id to be used (optional).
x25_if_name	string	Specifies the interface name to access. This is used in place of the device file name. A blank value means to use the first card configured.
x25_dev_name	string	Specifies the device name of the X.25 card. This is used in when WAN tracing is invoked at any layer. This is NOT used in place of x25_if_name. It is only for invoking tracing.

LAN Tests

Osidiag allows you to verify the local node can commnunicate with remote nodes and itself through the LAN link. These tests make direct use of the LAN product interface, so they do not test the OTS stack or services. Conversely however if you test the upper layers over LAN you will verify the LAN connectivity.

Test Menu

The test case menu displayed for the LAN tests is shown below:



```
OSIDIAG1.0                        802.3 Test Cases

        Highlight an item and then press "Return" or "Select Item".

                        Test Frames ...
                        Send Frame ...
                        Receive Frame ...
                        Close Device ...

                        Status ...

                        Utilities ->
```

Test Frames

This test will transmit an 802.2 "Test Frame" to the MAC address you specify. The link layer of the remote should echo back the data we sent on the test frame to us.

Success of this test indicates that the physical LAN connection as well as the LAN hardware of the local and remote are operational.

Failure of this test may be the result of the remote not supporting "Test Frames". Check the remote vendor's LAN link documentation to verify if this is the case. HP's link does support these frames.

Send Frames

This allows you to send data to the remote at the link layer. This operation requires knowledge of Network and Transport Layer PDU encodings to be successful.

HP does not recommend using this facility unless directed to do so by your HP support representative.

Receive Frames

This operation allows you to intercept PDU's from the remote which are intended for the OSI Network Layer.

This operation will fail if the OTS stack is up.

HP does not recommend using this facility unless directed to do so by your HP support representative.

Close Device

This will close the device file which was opened by executing the Send or Receive tests. It is not necessary for Test Frames.

Utilities

By selecting this item you will be taken to the Utilities menu. To return from the Utilities to the X.25 tests press Exit.

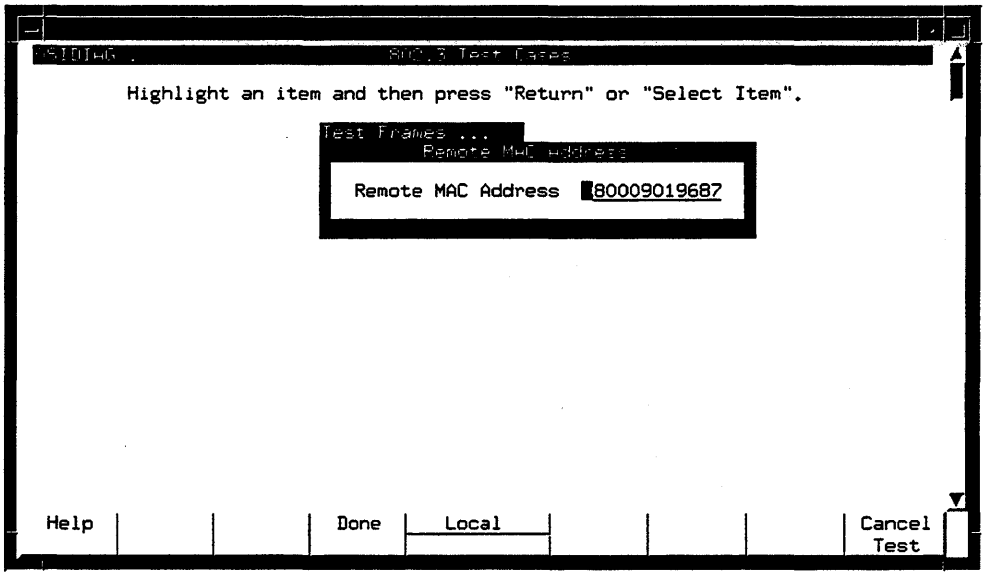
Status

This results in status and statistical information being retrieved from the LAN card and displayed. The UNIX utility lanscan(1M) is another good source of status information about LAN cards.

Parameters

The following subsections display the parameters you will be prompted for when running the various LAN tests.

Remote MAC Address



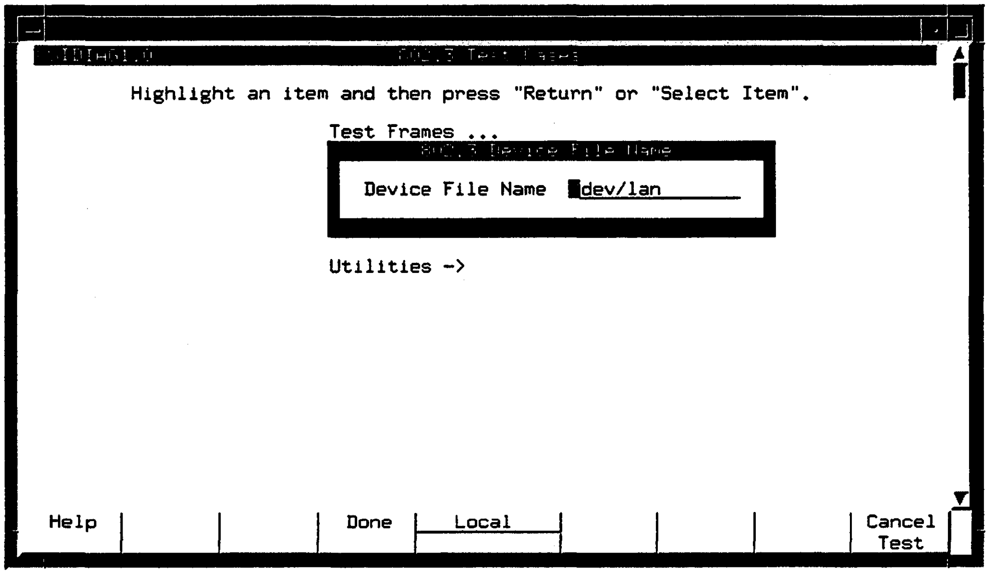
```
OSIDIAG .                    802.3 Test Cases

     Highlight an item and then press "Return" or "Select Item".

                      Test Frames ...
                               Remote MAC Address

                        Remote MAC Address  80009019687
```

This is the 6 byte Media Access Control Address of the remote LAN device which you will communicate with. This is also refered to as a **station address**.

By default the address of the local HP node is given.

To determine the MAC address of another HP node you can run the osidiag LAN status operation.

Local Device Name



```
OSIDIAG1.0                 802.3 Test Frames

        Highlight an item and then press "Return" or "Select Item".

                        Test Frames ...
                              802.3 Device File Name
                          Device File Name  dev/lan

                        Utilities ->
```

This is the device file which will be used for outbound communication. Its default is the configured device for the LAN subnet.

Example

The following shows a successful Test Frame operation.

```
entity:    LAN
test case: 802.2 Test Frames
------------------------------------
lan3_remote_addr    : 080009019687
lan3_dev_name       : /dev/lan
------------------------------------
open()                errno: 0
write()               errno: 0
read()                errno: 0
close()               errno: 0
------------------------------------
test status:  PASSED
=========================================================================
```

The first two lines indicate the test being executed. The second section of output indicates the parameters used to execute the test. The third section of output indicates the dialog which took place with the remote. The last section indicates the test status.

Parameter Display

The parameters used are described in Parameter Summary section which follows.

Dialog Display

As the test runs it displays its progress. The dialog is quite short.

- **open()** - opens the device file for access.
- **write()** - sends the test frame to the remote.
- **read()** - accepts the echoed test frame from the remote.
- **close()** - closes the device file.

Test Status

The status of the test is displayed. In this case the test passed.

Interpreting Errors

If a LAN test operation fails, the following output will appear in the status section at the end of the output:

```
-------------------------------------
test status:  FAILED
operation:    ioctl() - changing to TEST frames (super-user only)
return code:  1


EPERM:  Not super-user
=========================================================================
```

The meaning of each field is as follows:

- **test status** - indicates whether we passed or failed. Any error encountered during the operation (including setting up tracing and logging, if requested) will result in a status of failed.
- **operation** - indicates at what point in the test we encountered the first error. Typically this will be the name of file access operation, since the LAN driver is accessed as a raw device.
- **return code** - this is typically the value of the system errno returned on a system call. These are defined in the file "/usr/include/sys/errno.h" and in the manual page for errno(2).
- **Formatted error text** - after the above fields are displayed, osidiag will provide any textual information about the error. Typically this is displaying the meaning of the errno. In this case "EPERM: Not super-user". EPERM is the "#define" found in the errno.h file.

Further information on interpreting errors may be found in the Troubleshooting section of this manual.

Parameter Summary

The following table describes the parameters used for LAN tests.

For a description of the valid syntax for the various parameters see the section titled "Parameter Syntaxes".

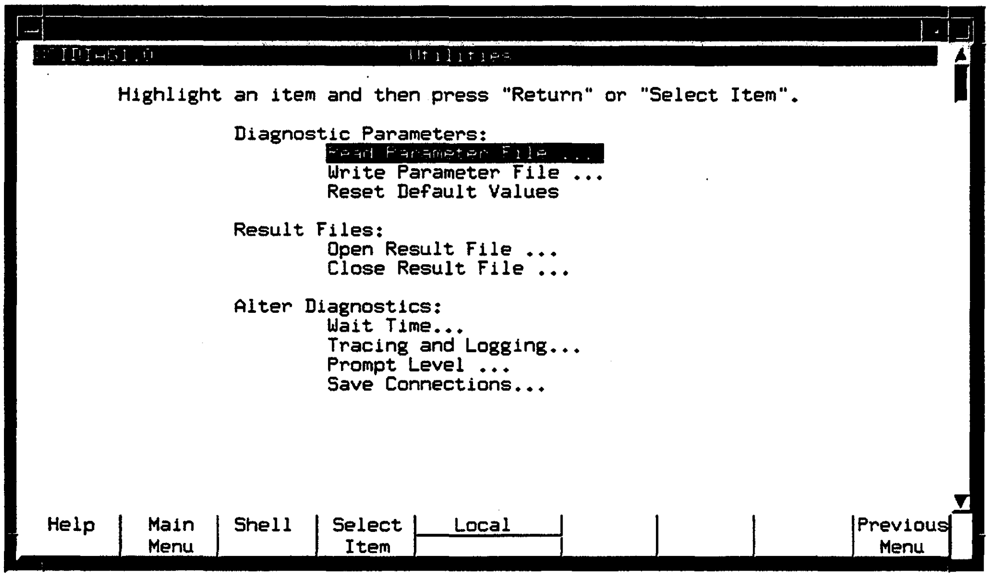
LAN Parameters		
parameter	**syntax**	**description**
lan_remote_addr	string	Specifies the remote MAC address.
lan_dev_name	string	Specifies the device file which we will use for LAN communication.
lan_data_file	string	Specifies a file which contains hex data to be carried on a 802.2 Unnumbered Information Frame. (Expert Only)
lan_data_val	hex	Specifies hex data to be carried on an 802.2 UI Frame. Use lan_data_file for large amounts of data.

Utilities

Osidiag provides several operations to enhance the default behavior of the tool.

Utilities Menu

The operations available are listed in the Utilities menu shown below. This menu can be accessed either from the main menu or from each of the test case menus.



```
IDI-51.0                                   Utilities

      Highlight an item and then press "Return" or "Select Item".

              Diagnostic Parameters:
                      Read Parameter File ...
                      Write Parameter File ...
                      Reset Default Values

              Result Files:
                      Open Result File ...
                      Close Result File ...

              Alter Diagnostics:
                      Wait Time...
                      Tracing and Logging...
                      Prompt Level ...
                      Save Connections...
```

Read Parameter File

This operation allows you to read a file containing values for any parameter you might be prompted for.

This is useful if you run osidiag against the same node on several occasions. You might have a parameter file which looks the following:

```
#
# Parameters for testing against node "hpindon"
#
tp_dst_tsap "diagt"."hpindon"
ss_dst_ssap "diags"."diags"."hpindon"
ut_pl_paddress N
```

The file shown provides destination TSAP and SSAP information, so that when running a Transport or Session connection test you won't have to type in the value. The last line changes the prompt level to not display the remote address prompt. This way when you select the connect test it will immediately begin running.

The parameter names are described in the "Parameter Summary" subsections for each layer (e.g. the parameters tp_xxx are described in the Transport section, the parameters ss_xxx are described in the Session section, etc.) The names of the parameters being used also appear at the top of the output for a test.

When reading in a parameter file all parameter values which are not mentioned retain their original values.

You may read in more than one parameter file if desired.

You may specify a parameter file to be read in from the command line as well, by using the "-p" option.

Parameter files can be created manually through an editor, or you may generate one from the values you are currently using by using the "Write Parameter File" utility.

Write Parameter File

This operation will dump out all the current parameter values which have non-null values for the last entity you tested. If you have not run any tests through osidiag, only the utility parameters with non-NULL values will be written out.

If you have modified any parameter values from their default values these changes will be reflected.

The resulting file is compatible with the form used by the "Read Parameter File" operation.

Osidiag will overwrite (not append) this information to the file you specify.

Reset Default Values

This operation will set all parameters to their default values (i.e. configured for loopback to the local node). Any modifications which you have made prior to this command will be undone.

Open Result File

Ordinarily when you run osidiag the results of a test case are erased once you return to the test menu.

By opening a result file, a duplicate of the information displayed will be created in the file you specify.

The resulting file will be useful in providing information to your HP support respresentative in the case of problems. If you have enabled any tracing or logging this will also be written to the file.

Close Result File

Once a result file is opened it will record all of the test activity until you exit osidiag or explicitly close it by selecting this operation.

Wait Time

This controls the amount of time a test will wait for an indication (or confirmation). If the time elapses and no indications are received then the test case will exit with an error.

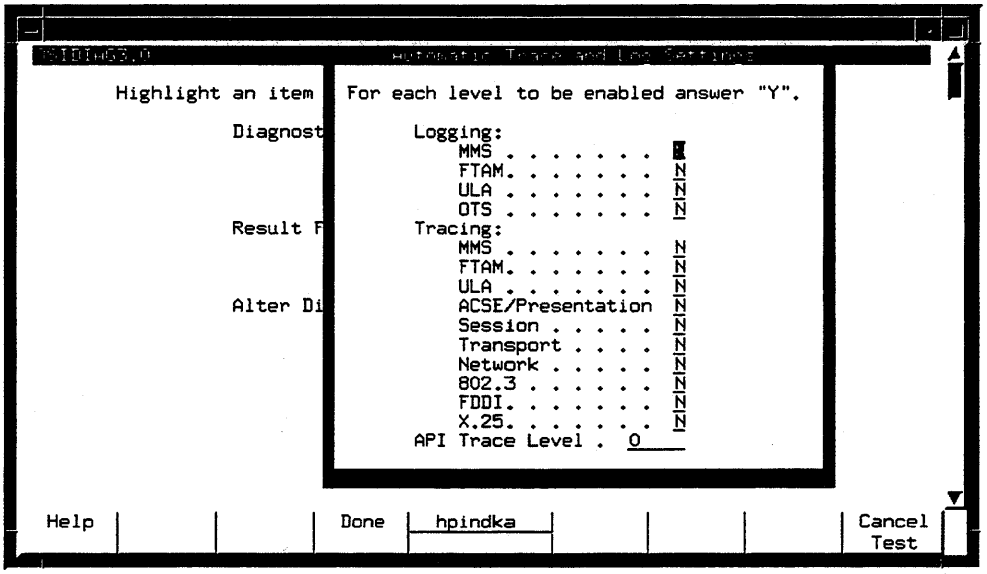
The default value is 30 seconds.

Tracing and Logging

An important feature of osidiag is the ability to automatically enable the tracing and logging facilities at the various layers. The nettl(1M) and netfmt(1M) are used to control tracing and logging at the various layers.

API tracing can also be enabled through the programmatic interface for those services supporting it: FTAM, ROSE, APLI, Session and Transport.

When you select Tracing and Logging from the menu you will be presented with the following screen:



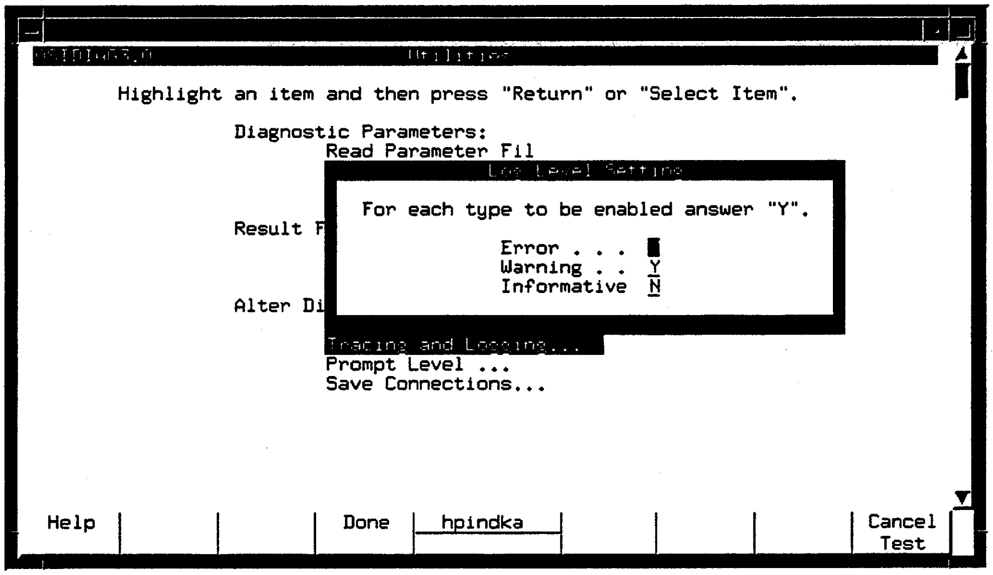
```
For each level to be enabled answer "Y".

     Logging:
         MMS . . . . . . . . █
         FTAM. . . . . . . . N
         ULA . . . . . . . . N
         OTS . . . . . . . . N
     Tracing:
         MMS . . . . . . . . N
         FTAM. . . . . . . . N
         ULA . . . . . . . . N
         ACSE/Presentation   N
         Session . . . . . . N
         Transport . . . . . N
         Network . . . . . . N
         802.3 . . . . . . . N
         FDDI. . . . . . . . N
         X.25. . . . . . . . N
     API Trace Level . 0
```

- **MMS Logging** - this enables the formatting of MMS entity log messages.
- **FTAM Logging** - this enables the formatting of log messages for the four FTAM entities: FTAM_INIT, FTAM_RESP, FTAM_VFS and FTAM_USER.
- **ULA Logging** - this enables the formatting log messages for the various ULA components which support both FTAM and MMS. These components are described in more detail in the "Logging and Tracing" chapter of this manual.
- **OTS Logging** - this enables formatting of log messages for any of the OTS entities. See the "Logging and Tracing" chapter for discussion of these entities.
- **MMS Tracing** - this enables tracing at the MMS layer. MMS only displays PDU header information. Use ULA Tracing to see PDU contents.
- **FTAM Tracing** - this enables tracing at the FTAM layer. This can show the FTAM PDUs being sent and received.

- **ULA Tracing** - ULA provides the ACSE/Presentation service for FTAM and MMS. Tracing here will provide more information about the connection dialog.
- **ACSE/Presentation Tracing** - this enables tracing at the ACSE/Presentation service provided by OTS. NOTE: this service is only used by APRI applications. The Application layer services (i.e. FTAM, MMS, X.400, etc.) do not use OTS to provide this. Appeared at 8.0
- **Transport layer tracing** - this enables tracing at the Transport layer. All Transport traffic taking place during the test run will be traced.
- **Network layer tracing** - this enables tracing at both the CLNS and CONS entities. This is the lowest level of tracing done by OTS.
- **LAN Tracing** - this enables tracing at the LAN driver. Osidiag will automatically filter out non-OSI traffic by restricting the SSAP and DSAP to the OSI value 0xFE.
- **FDDI Tracing** - this enables tracing at the FDDI driver.
- **X.25 Tracing** - this enables tracing at the X.25 driver. You can only trace on one X.25 card at a time.
- **API Trace Level** - specifying a non-zero value enables API tracing. The exact value you specify should be the decimal representation of the API trace flags to enable. The value 7 enables basic API tracing. See the Programmer's Guide for the particular service to determine what API trace flags are available.

Log Level

As discussed in the "Logging and Tracing" chapter, the level of logging performed can be controlled. When any logging is selected from the Tracing and Logging screen, you will be presented with this screen to select the level of logging.

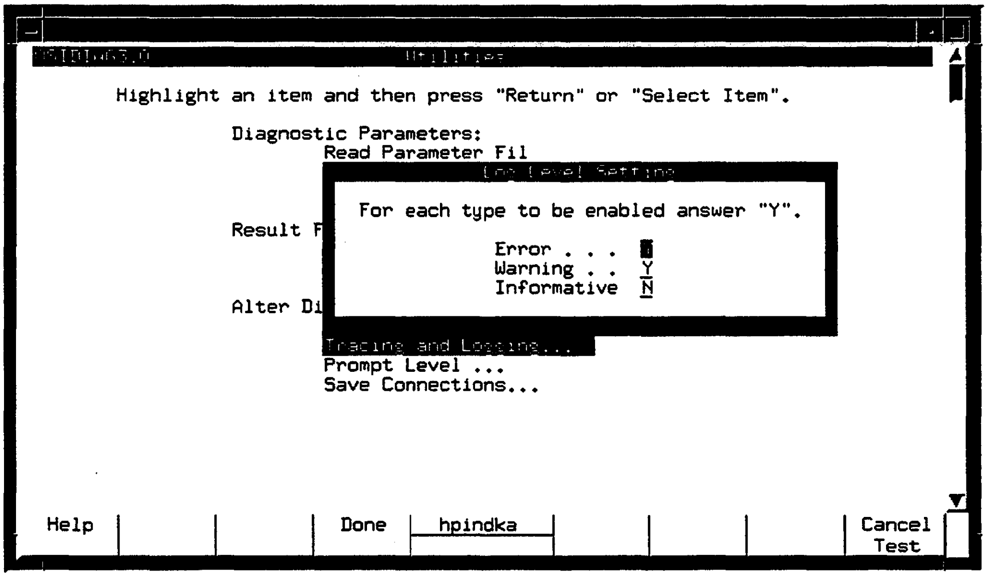


```
OSIDIAG3.0                       Utilities

       Highlight an item and then press "Return" or "Select Item".

                 Diagnostic Parameters:
                      Read Parameter Fil
                      ┌──────────── Log Level Setting ────────────┐
                      │                                           │
                      │  For each type to be enabled answer "Y".  │
                 Result F                                         │
                      │           Error . . . █                   │
                      │           Warning . . Y                   │
                      │           Informative N                   │
                 Alter Di                                         │
                      └───────────────────────────────────────────┘
                      Tracing and Logging...
                      Prompt Level ...
                      Save Connections...

 Help |    |    | Done | hpindka |    |    |    | Cancel
                                                    Test
```

Note that the default values are shown, which include ERROR and WARNING log messages. Adding INFORMATIVE logging may greatly increase the amount of information logged.

Trace Type

As discussed in the "Logging and Tracing" chapter, there are several kinds of tracing information which the entities are capable of producing. When any tracing is enabled, you will be presented with the following screen to select the kind of tracing to be performed.



```
OSIDIaG3.0                          Utilities

      Highlight an item and then press "Return" or "Select Item".

                Diagnostic Parameters:
                       Read Parameter Fil
                                      Log Level Setting

                            For each type to be enabled answer "Y".
                Result F
                                       Error . . .  █
                                       Warning . .  Y
                                       Informative  N
                Alter Di

                       Tracing and Logging...
                       Prompt Level ...
                       Save Connections...
```

Note that the default values are shown, which are inbound and outbound PDUs and Headers. These kinds of tracing will show information about the protocol information being sent out and received from the remote.

The other types of tracing will increase the amount of trace information displayed. See the "Logging and Tracing" chapter for more information about what type of information the various trace kinds produce.

Example

Tracing will take effect when a test case is run and the formated trace output will be appended at the end of the test output. The following shows log and trace output as it appears at the end of an osidiag test case. This particular output was created by enabling OTS logging and TRANSPORT tracing, using the default log level and trace kind. The Transport Connect test was then run with an incorrect destination address. Data was also specified to be carried on the outbound connect request (non-default behavior).

```
------------------------------------
test status:  FAILED
operation:    t_rcvdis()
t_errno:      13
errno:        2


osidiag:  Received unexpected disconnect.
     Examine the reported reason code for more information.  This may
     indicate that the address specified was incorrect.  Check the status
     of the remote entity for further information.
------------------------------------
Formatting log file:  /usr/adm/nettl.LOG00
Initializing formatter....

Initialization done.

***********************************HP OSI MGMT******************************@#%
  Timestamp              : Mon Jan 27 1992 10:13:27.099298
  Process ID             : ICS:              Subsystem        : TRANSPORT
  User ID ( UID )        : -1                Log Class        : ERROR
  Device ID              : -1                Path ID          : -1
  Connection ID          : -1                Log Instance     : 1245

[4113] Routing: tr4cstz.c 1048
[....] Command=81e0 Nsap=6870696e646b61 Nsap=6870696e7a7a61 Parm=(1 0)

***********************************HP OSI MGMT******************************@#%
  Timestamp              : Mon Jan 27 1992 10:13:27.119540
  Process ID             : ICS:              Subsystem        : TRANSPORT
  User ID ( UID )        : -1                Log Class        : ERROR
  Device ID              : -1                Path ID          : -1
  Connection ID          : -1                Log Instance     : 1246

[4108] T_REJECT tr4sub.c 322
[....] ENT=41 CTX=1 CHN=1 SAP=254 SUF=056469616774 SUF=056469616774 SAP=70
[....] ADD=00 ADD=076870696e7a7a61 CTX=0 CHN=0 PDS=4096 REF=8401 REF=0000
[....] Status=03430000 Cause=0000f303
------------------------------------
```

```
Formatting trace file:  /tmp/osa02653.TRC0
Initializing formatter....

Initialization done.

vvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvHP OSI MGMTvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvv@#%
  Timestamp             : Mon Jan 27 1992 10:13:27.078254
  Process ID            : ICS:                Subsystem         : TRANSPORT
  User ID ( UID )       : -1                  Trace Kind        : HDR OUT TRACE
  Device ID             : -1                  Path ID           : -1
  Connection ID         : -1

[9973] genmai2.c 1643
[....] T-CONNECT-Req Buffer1 is User Data Local Sap=000000fe Size=00000000
[....]  Class/Options=61 Qos=00 Remote T-address=056469616774 ..=
[....] 076870696e7a7a61 Local T-address=00 ..=056469616774

vvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvHP OSI MGMTvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvv@#%
  Timestamp             : Mon Jan 27 1992 14:09:53.545303
  Process ID            : ICS:                Subsystem         : TRANSPORT
  User ID ( UID )       : -1                  Trace Kind        : PDU OUT TRACE
  Device ID             : -1                  Path ID           : -1
  Connection ID         : -1

[9974] genmain.c 1575
[....] Total Buffer Length: 15
[....] Buffer: 1  Subchain:  0
[....] 64 61 74 61 20 6F 6E 20 63 6F 6E 6E 65 63 74     # data on connect

^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^HP OSI MGMT^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^@#%
  Timestamp             : Mon Jan 27 1992 10:13:27.133819
  Process ID            : ICS:                Subsystem         : TRANSPORT
  User ID ( UID )       : -1                  Trace Kind        : HDR IN TRACE
  Device ID             : -1                  Path ID           : -1
  Connection ID         : -1

[9973] genmai2.c 1643
[....] T-DISCONNECT-Ind Buffer1 is User Data Reason=f3
==========================================================================
```

Note that osidiag will indicate what is being formatted in the output. In this case "/usr/adm/nettl.LOG00" is the log file being formatted, and "/tmp/osa02653.TRC0" is the name of the trace file. Osidiag will delete this temporary trace file at the end of the test.

For more information on interpreting the log and trace messages, see the "Logging and Tracing" chapter, the "Common Logged Errors" section of the "Troubleshooting" chapter, and the "Messages" chapter.

If no log or trace information is present, osidiag will still report that it is invoking the formatter, you will just see no formatted records after the phrase "Initialization done."

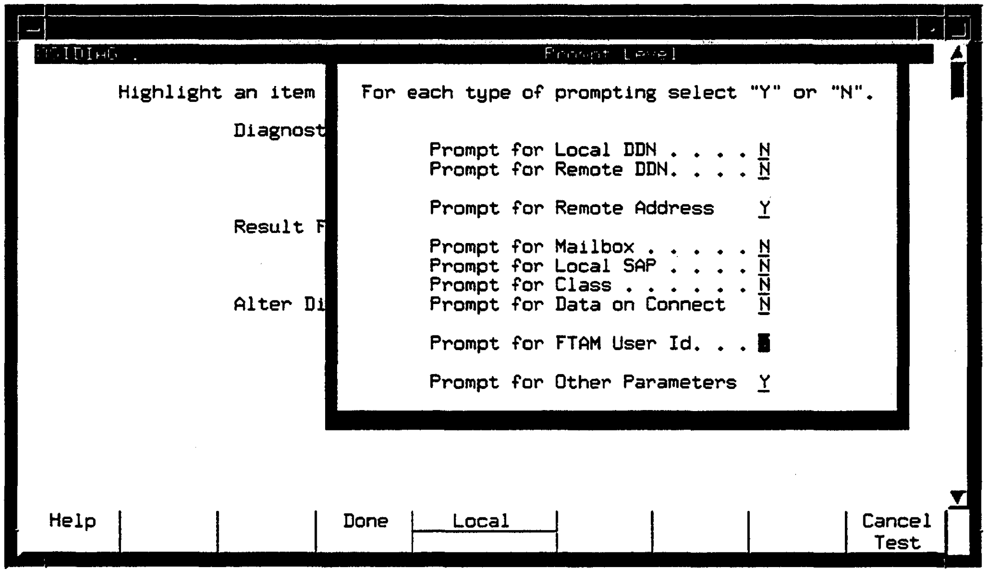
Prompt Level

This utility allows you to control what parameters osidiag explicitly asks for and which ones will take on their default values without prompting.

You may wish to turn prompting off for a class of parameters if you are satisfied with the current value. An example would be if you were running the connection test against a node multiple times. Rather than having the "Destination TSAP" form displayed each time you invoke the connection test, you can turn off prompting for remote address. Then the next time you run the connect test it will skip the display of the form and begin immediately, using the TSAP value you had specified previously.

You may wish to turn prompting on for a class of parameters if you want to override the defaults. For example the transport tests by default do not send any data on the connection. Furthermore when you run a connect test you are not asked if you want data to be carried. By turning on prompting for data on connection, the next time you run the connect test you will be prompted for connection data.

The following form is displayed to allow you to alter the prompt level.



```
For each type of prompting select "Y" or "N".

     Prompt for Local DDN . . . . N
     Prompt for Remote DDN. . . . N

     Prompt for Remote Address    Y

     Prompt for Mailbox . . . . . N
     Prompt for Local SAP . . . . N
     Prompt for Class . . . . . . N
     Prompt for Data on Connect   N

     Prompt for FTAM User Id. . . █

     Prompt for Other Parameters  Y
```

- **Local DDN** - this parameter is used for FTAM and MMS tests. By default you are not asked for the Local DDN. This value is extracted automatically from the configuration file. If you wish to perform tests over a different local DDN, then set this to "Y".
- **Remote Alias** - this parameter is used only for FTAM tests. The remote alias is the value you configured for the remote FTAM entities which you can access. This value is actually an abbreviated version of the remote FTAM directory name. A full DDN is created by substituting the alias into the ddn_lookup_path parameter configured in

the local_app configuration file in "/etc/net/osi/conf".

- **Remote DDN** - this parameter is used for FTAM and MMS tests. The remote DDN is the symbolic name for the remote Responder with which you want to communicate.
- **Remote Address** - controls whether you will be asked for the **destination Presentation Address** for **FTAM** and **MMS** tests, the **destination Session Address** for **X.400** and **Session** tests, the **destination Transport Address** for **Transport** tests, the **destination X.121 Address** for **X.25** tests and the **MAC Address** for **LAN** tests. By default this is enabled.
- **Special Files** - by default you are not asked for the mailbox name and typically should not need to change this field unless you run into naming conflicts. This applies to Session and X.400 tests only.
- **Local SAP** - by default you are not asked for this and typically should use the defaults. One instance where you may need to modify the default is when running the Transport or Session Server tests on a local node and intercepting X.400 traffic from a remote. See the section on "Testing Between Different Layers" for more details.
- **Class, Version** - by default the Transport Class proposed will be 4 for LAN tests and 2 for WAN tests and you will not be prompted for a different value. By setting this to "Y" you have the option of proposing class 0, or proposing class 2 with an alternate of class 0 when running WAN tests.

  This also controls prompting for the MMS version.
- **Data on Connect** - by default no data is carried on the APRI, Session or Transport connect or disconnect requests or responses. By setting this to "Y" you may specify data to be carried on these PDU's.
- **FTAM User ID** - the user id identifies yourself to the remote FTAM responder. If you have already specified these and want to continue using the same information set this to "N".
- **Other Parameters** - all the parameters which are not covered by the previous categories.

Save Connections

This operation allows you to modify the behavior of the MMS, CMIP, APRI, Transport or Session tests.

A typical test case goes through the following 3 phases:

- **Connection Establishment** - any registration or other activity takes place to prepare us locally for a connection and then we either send or listen for a connection. When the connection has been confirmed we have completed this phase.
- **Activity Over Connection** - after we have an established connection we may optionally perform some activity over that connection. For example send data or receive an indication.
- **Connection Tear Down** - when the test has completed (or failed) with the connection and any other activity the connection and association with the stack are released.

When the "Save Connections" option is chosen the last phase (connection teardown) is only performed in case of an error. If a subsequent test case is requested only the second step is performed. The "Disconnect" operation under the test case menu overrides this and shuts the connection.

A common use for this is to send data multiple times, or to receive an indication and then send data. A sample dialog in save connection mode is shown below:

Local	Remote	NOTES
Connect	Server	Connection established control returned to test case menu.
Data	Server	Data sent from local received by remote
Server	Data	Data sent from remote received by local
Disconnect	Server	Disconnect initiated by local received by the remote

Utility Parameters

The following table summarizes the Utiltity parameters which are configurable by the caller of osidiag.

Utility Parameters		
parameter	**syntax**	**description**
ut_pl_paddress	yes_no	indicates whether the user will be prompted for the remote presentation address or subset thereof (i.e. TSAP) when a test case is run.
ut_pl_local	yes_no	indicates whether the user will be prompted for the local DDN when a test case is run. MMS and FTAM only.
ut_pl_remote	yes_no	indicates whether the user will be prompted for the remote DDN when a test case is run. MMS and FTAM only.
ut_pl_alias	yes_no	indicates whether the user will be prompted for the remote FTAM alias when a test case is run.
ut_pl_ftamid	yes_no	indicates whether the user will be prompted for FTAM user id information when a test case is run.
ut_pl_mbox	yes_no	indicates whether the user will be prompted for the name of the pipe file. Obsolete.
ut_pl_local_sap	yes_no	indicates whether the user will be prompted for the local address. (Use ut_pl_local for FTAM and MMS).
ut_pl_class	yes_no	indicates whether the user will be prompted for the Transport class and MMS version number.
ut_pl_call_data	yes_no	indicates whether the user will be prompted for data on connect and disconnect PDU's.
ut_pl_other	yes_no	indicates whether the user will be prompted for the other common parameters not covered by other categories. These are the non-addressing parameters you are prompted for by default.

Utility Parameters (continued)		
parameter	**syntax**	**description**
ut_mms_log	yes_no	indicates if MMS logging should be formatted.
ut_ftam_log	yes_no	indicates if FTAM logging should be formatted.
ut_ula_log	yes_no	indicates if ULA logging should be formatted.
ut_ots_log	yes_no	indicates if OTS stack logging should be formatted.
ut_mms_tr	yes_no	indicates if MMS tracing should be enabled.
ut_ftam_tr	yes_no	indicates if FTAM tracing should be enabled.
ut_ula_tr	yes_no	indicates if ULA tracing should be enabled. Use for FTAM and MMS.
ut_ap_tr	yes_no	indicates if ACSE/Presentation tracing should be enabled. For APRI only. Use ULA for FTAM and MMS.
ut_sess_tr	yes_no	indicates if Session Layer tracing should be enabled.
ut_tran_tr	yes_no	indicates if Transport Layer tracing should be enabled.
ut_net_tr	yes_no	indicates if Network Layer tracing should be enabled.
ut_lan_tr	yes_no	indicates if LAN link tracing should be enabled.
ut_fddi_tr	yes_no	indicates if FDDI link tracing should be enabled.
ut_wan_tr	yes_no	indicates if WAN link tracing should be enabled.

Utility Parameters (continued)		
parameter	**syntax**	**description**
ut_mm_svc	yes_no	indicates whether the MMS connection will be saved across test cases.
ut_cmis_svc	yes_no	indicates whether the CMIS connection will be saved across test cases.
ut_apri_svc	yes_no	indicates whether the APLI or ROSE connection will be saved across test cases.
ut_ss_svc	yes_no	indicates whether the Session connection will be saved across test cases.
ut_tp_svc	yes_no	indicates whether the Transport connection will be saved across test cases.
ut_wait_time	integer	specifies the number of seconds to wait for an indication or confirmation before timing out. Overrides "-w" on command line.
ut_api_level	integer	Gives the level of API tracing to be performed. This is the decimal value for the API flags.
ut_log_error	yes_no	indicates if the error log level is enabled.
ut_log_warn	yes_no	indicates if the warning log level is enabled.
ut_log_info	yes_no	indicates if the informative log level is enabled.
ut_tr_pdu_in	yes_no	indicates if Inbound PDU tracing is enabled.
ut_tr_pdu_out	yes_no	indicates if Outbound PDU tracing is enabled.
ut_tr_hdr_in	yes_no	indicates if Inbound PDU header tracing is enabled.
ut_tr_hdr_out	yes_no	indicates if Outbound PDU header tracing is enabled.
ut_tr_proc	yes_no	indicates if Procedure tracing is enabled.
ut_tr_state	yes_no	indicates if State tracing is enabled.
ut_tr_error	yes_no	indicates if error tracing is enabled.
ut_tr_log	yes_no	indicates if logged message tracing is enabled.

Testing Between Different Layers

This section attempts to address the issue of running tests against another system when there is not an exact matching of tests. It turns out that even if the other vendor has no tools like osidiag on their equipment we can still often run tests at the various layers of the stack.

Connection Dialog

The figure below shows the exchange of Protocol Data Units (PDUs) in a typical connection test between the RTS connect test (initiator) and an RTS server test. This exact dialog would take place when running Transport Class 2 over X.25 (the differences for other links and Transport classes is discussed in the next section).

The scenario for FTAM is very similar, the main difference being that upper layer data is carried on the Session release as well as connection.

```
  INITIATOR                         SERVER
  =========                         ======

+-------+                      1.
| CALL  |------------------------>
+-------+

                               2.   +---------+
          <---------------------| ACCEPT |
                                    +---------+

+------+-----+                 3.
| DATA | CR  |------------------>
+------+-----+

                               4.   +------+-----+
          <---------------------| DATA | CC |
                                    +------+-----+

+------+----+----+------+      5.
| DATA | DT | CN | Conn |------->
+------+----+----+(upper layer info)
                               6.   +------+----+----+--------+
          <---------------------| DATA | DT | AC | Accept |
                                    +------+----+----+--------+

+------+----+----+             7.
| DATA | DT | FN |-------------->
+------+----+----+

                               8.   +------+----+----+
          <---------------------| DATA | DT | DN |
                                    +------+----+----+

+------+-----+                 9.
| DATA | DR  |------------------>
+------+-----+

                              10.   +------+-----+
          <---------------------| DATA | DC |
                                    +------+-----+

+--------+                    11.
| CLEAR  |------------------------>
+--------+
```

lan misses these

osidiagtest connect skips to 9.

Each message shown corresponds to a Network layer packet which may carry as data PDUs from the upper levels. The vertical lines breaking up the packets indicate the separation between the layers. The first field is the Network PDU, the second field the Transport PDU, the third field the Session PDU and the fourth field is the RTS PDU (note the RTS plays the role of both the Presentation and Application layer in the OSI Reference model).

Description of PDU's

The PDU types shown are as follows:

- **CALL** - this is an X.25 request to establish a connection.
- **ACCEPT** - this is an X.25 positive response to a CALL.
- **DATA** - this is data being carried on an X.25 connection.
- **CR** - this is a Transport request to establish a connection (Connect Request).
- **CC** - this is a positive response to a connection request (Connect Confirm).
- **DT** - this is data being carried on a Transport connection.
- **CN** - this is a Session layer request for a connection.
- **AC** - this is a Session layer acceptance of a connection.
- **Connect** - this represents the connect request at the layers above Session.
- **Accept** - this represents the connect acceptance at the layers above Session.
- **FN** - this is a Session layer Finish PDU indicating that we are done with the connection.
- **DN** - this is a Session layer Disconnect PDU releasing the connection in response to the FN.
- **DR** - this is a Transport layer Disconnect Request.
- **DC** - this is a Transport layer Disconnect Confirm in response to the DR.
- **CLEAR** - this ends an X.25 connection.

Description of Dialog

From the diagram we can see that connection establishment takes place in three distinct phases. Likewise the connection release takes place in three phases.

Messages **1** and **2** establish the X.25 connection. For tests run over LAN these messages are not sent since LAN uses the Connectionless Network Service.

Messages **3** and **4** establish the Transport connection. The connection dialog between the two transport entities is carried as data on the previously established Network connection. For connectionless the dialog is sent over Network datagrams.

Messages **5** and **6** establish the Session connection as well as the connections for the layers above session (in this case RTS). The Session (and upper layer) connection dialog is carried as data on the previously established Transport connection.

Messages **7** and **8** tear down the Session connection and the connections for the layers above. In the case of RTS the connection release is implicit (no PDU is sent). The release dialog is carried over Transport data.

Messages **9** and **10** disconnect at the Transport level. This dialog is carried over Network data. For Transport Class 0 this dialog does not take place rather the disconnect is implied by message 11.

Message **11** releases the X.25 connection. For connectionless this message is not sent.

Different Layer Testing as Initiator

Because of the three tiered nature of connection establishment we can easily run the X.25 and Transport connection tests against a remote's RTS instead of a remote X.25 or Transport server. The layer specific servers are preferable because they will tend to provide more information in case of an error, but they are not required. This ability is useful if no X.25 or Transport specific tests exist on the remote.

The reason we can run the X.25 or Transport connect tests against an RTS is that no upper level information is required on the connect call. Session and RTS (and FTAM) PDU's aren't sent until after these lower layer connections are established.

So a Trasnport connect test run against a remote RTS (or FTAM Responder) would result in the following sequence of messages: **1, 2, 3, 4, 9, 10, 11.** For Transport class 0 we omit messages 9 and 10 and for LAN we omit messages 1, 2 and 11. In any of these cases however we have generated a valid sequence of calls and if both stacks are operating correctly we should anticipate no errors.

Likewise an X.25 connect test run against a remote RTS would result in the message sequence: **1, 2, 11.**

We can certainly run the RTS connection test against the remote RTS (or the FTAM connect test against the FTAM Responder), but what about a Session connection test? The Session connection test will fail because by default it carries no data on the connection, and on the remote RTS side, an RTS connect PDU is expected on the Session CN. Likewise FTAM expects upper layer PDU's.

In fact the X.400 RTS tests provided by osidiag basically run the Session connection test, but specify data (specifically an RTS PDU) to be carried on the Session CN.

Different Layer Testing as Server

Conversely what happens if we are running our server tests and the remote is generating network traffic from the RTS (or FTAM Initiator)? In this case we'll be able to observe that we can receive and accept their connection request, however after sending the request the remote will then proceed to send a conection PDU for a higher level as data to our server. Because our server has no knowledge about the higher level protocol we will not generate the expected response. This also holds for an FTAM Responder.

X.25

If we run the X.25 server and generate traffic from the remote RTS (or FTAM Initiator) then we should receive message **1** and the server will send message **2.** Now that an X.25 connection is established the remote will send message **3,** but our X.25 server will not send message **4** because it views the received packet solely as data. What the X.25 server does do when it receives data is release the connection.

Although the X.25 server dialog results in an error for the remote Trasnport entity, it does demonstrate the the ability to receive X.25 connections, which is the purpose of the X.25 server.

Transport

If we run the Transport server test and are connecting to it from a remote RTS (or FTAM Initiator) then one issue which did not exist for the X.25 server is addressing. If we direct the remote side's X.400 system to send mail to our local node the address used will be that configured for the local node's RTS not for that of the Transport server.

OTS does not permit us to listen at RTS's address at the Transport level, so we must modify the address configured for our RTS in the remote system's routing table to match the address listened on by the Transport server.

The default address our server listens on is 6469616774 ("diagt").

When we run the Transport server and generate traffic from the remote RTS then we should receive message **3** from the remote and will respond with message **4.** After this Transport connection dialog completes the remote will attempt to establish the upper layer connections by sending message **5.** The server will receive this Session CN PDU as Transport data. The server will display the data received and listen for another indication.

Note that at this point the Transport server is waiting for an indication and the remote RTS is waiting for a confirmation for its Session layer connect. What will typically happen at this point is that the Transport server will timeout after 30 seconds and disconnect.

Again as in the case of the X.25 server test we encounter some errors in the test, but we have shown that a Transport connection can be brought up by the remote and data can be sent on it.

Session

As with the HP as initiator tests, the Session server test will fail because it will not carry the RTS connect response PDU on the Session accept PDU. When running the Session test however we can listen on the address used by FTAM or RTS. You must first change the "Prompt Level" as described in the "Utilities" section.

Parameter Syntaxes

The following subsections summarize the syntax for the various parameter types.

Integer

This parameter is simply a signed decimal value.

String

String parameter types are simply character strings. Blanks may not be included in these strings.

Yes_No

For parameters which act as switches the values may be either **Y** or **N.** This is case sensitive.

X.121

An X.121 Address is entered as a decimal value. The maximum length is 15 digits long. An example is:

```
  4084472000
```

MAC

A MAC address is entered as a hexadecimal value. It must be exactly 12 digits long. An example value is:

```
  080250001234
```

Hex

A hex parameter is a string of hex digits of even length. An example hex entry for both a parameter file or a form is:

```
  A60602010F02010C
```

Osidiag also allows you to specify values in ASCII by using quotation marks. For example

```
  "diags"
```

SSAP

SSAP's are comprised of three hex values, the Session Selector, the Transport Selector and the Network Address. These byte strings should be specified in hex format.

When specifying an SSAP in a parameter file the selectors should be separated by a period. An example parameter file entry would be:

```
3330.4441.6867696e646f6e01
```

Values which are ASCII may also be specified as quoted strings.

TSAP

TSAP's are comprised of two hex values, the Transport Selector and the Network Address. These values should be specified in hex format.

When specifying an TSAP in a parameter file the fields should be separated by a period. An example parameter file entry would be:

```
4441.6867696e646f6e01
```

Values which are ASCII may also be specified as quoted strings.

Paddr

A presentation address is comprised of four components, the Presentation Selector, Session Selector, Transport Selector and Network Address. All components should be specified as hex values, each separated by a blank. To specify a NULL for one of the selectors leave it blank, but maintain the period separators.

For example:

```
01ff..231a.4900367788
```

Indicates a Presentation Address with a NULL Session Selector.

Local addresses for Session and Transport addresses are subsets of the Presentation address format.

For a local Session address we specify the Session selector and the Transport selector. The Network address is unnecessary because it will always be the value configured for our node. An example value would be:

```
"diags"."diags"
```

For a local Transport address we specify only the Transport selector. The Network address is unnecessary because a NULL Network address tells XTI to listen on both

CONS and CLNS. An example value would be:

```
"diagt"
```

These values can be specified in hexadecimal or as quoted ASCII strings.

NOTE: if you choose to specify the local Transport including the Network Address, you must indicate that the unused P- and S-selectors by using the periods as place holders. Otherwise the value will be interpreted as a Session Address.

DDN

A Directory Distinguished name is made up of the following attribute types:

- **C** - a country code, optional and may only appear once.
- **L** - the name of a locality, for example a city name "cupertino".
- **O** - an orginization name, mandatory and may only appear once.
- **OU** - an orginizational unit, optional and may appear several times.
- **AP** - application title, mandatory and may only appear once.
- **AE** - application entity, mandatory and may only appear once.

When specifying a DDN we use the following syntax:

```
/C=US/O=hp/OU=hpindon/AP=ftam/AE=resp
```

The attributes are separated with the "/" character and each attribute name is immediately followed by the "=" character and its value.

PCDL

A Presentation Context Definition List (PCDL) is used for the APRI test cases to specify the abstract syntaxes (protocols) and transfer syntaxes (usually ASN.1) to be used for the duration of a connection. Each context definition also has a Presentation Context Identifier (PCI) which allows subsequent references to the context to use a single numeric value.

The form of a PCDL is defined by the following grammar:

```
pcdl:
  { context_list }

context_list:
  NULL
  context_list context_elt
```

```
context_elt:
  { pci objid { objid_list } }

pci:
  INTEGER

objid:
  { integer_list }   /* where integer_list length is >= 2 */

objid_list:
  NULL
  objid_list objid

integer_list:
  INTEGER
  integer_list INTEGER
```

An example PCDL in this format is:

```
{ {3 {2 2 1 0 1} {{2 1 1}} }  {5 {2 1 9999 1} {{2 1 1}} } }
```

IntList

An integer list is simply a list of integers. It should have a non-zero length, as formally defined in the grammar for PCDL.

ObjId

Object Identifiers are widely used in OSI for Layers 6 and 7. Their main purpose is as a mechanism to guarantee unique identifiers. This is done by making the identifier a sequence of numbers, where the initial sequence is controlled by some standards body. See the ISO 8824 Annex B for more information.

In osidiag object identifiers should follow the definition given in the grammar for PCDL's.

An example object identifier is:

```
{2 1 1}
```

RefId

A Reference Id is a value which may either be an Object Id or a single integer value. When passed as arguments to an API they typically are passed as unions, where the integer value is refered to as a Local value and the object id as a Global.

When specified through osidiag the following grammar defines it:

```
refid:
  INTEGER
  objid
```

MIB

Osidiag allows you to alter the default Management Information Base (MIB) described in the CMIS section. This can only be done through parameter files.

The following grammar describes the structure of a MIB. The character "\" may be used to extend the MIB definition across multiple lines for greater legibilty. It should immediately proceed the linefeed character.

```
mib:
  { obj_instance_list }

obj_instance_list:
  NULL
  obj_instance_list obj_instance

obj_instance:
  ( obj_class ( attr_list ) { child_obj_list } )

obj_class:
  refid

attr_list:
  NULL
  attr_list attr_elt

attr_elt:
  ( attr_id attr_type attr_val )

attr_id:
  refid

attr_type:
  O  |  P  |  R  |  I    /* As defined in CMIS section */

attr_val:       /* interpretation depends on preceding <attr_type> */
  objid
  STRING
  HEXSTRING
  INTEGER

child_obj_list:
    obj_instance_list
```

An example value is the default CMIS MIB:

```
   { (\
       {1 2 124 36051 3 1} \
       (\
         (\
           {1 2 124 36051 1 20} \
           0 {2 1 9999 1}\
         ) \
       ) \
#  The child objects begin here.  There is only one       {\
         (\
           {1 2 124 36051 3 17} \
           (\
             (\
               {1 2 124 36051 1 143} \
               P osidiag\
             ) \
             (\
               {1 2 124 36051 1 144} \
               I 0\
             ) \
           ) \
         ) \
       }\
   ) }
```

Attr_Val

CMIS uses Attribute Value lists two ways. First to identify object instances. And secondly to specify attributes to be modified in a particular object instance.

The grammar builds off that defined for the MIB above and is as follows:

```
  attr_val_list:
    ( attr_list )
```

An example attribute value list would be:

```
  ( ( 3121 R 020377 ) )
```

Attr_Id

CMIS uses Attribute Id lists to identify attributes to be retrieved from a particular object instance.

The grammar builds off that defined for the MIB above and is as follows:

```
attr_id_list:
  ( attr_ids )

attr_ids:
  NULL
  attr_ids attr_id
```

An example attribute id list would be:

```
( {2 1 777 6} {2 1 777 4} )
```

8

Messages

Messages that can be encountered while using Hewlett-Packard's OSI products

Messages

This chapter contains messages that can be encountered while using Hewlett-Packard's OSI products. See the other OSI product manuals (listed at the end of this manual) for additional error messages.

OSIADMIN Error Messages

MESSAGE	`stat_file: Can't stat XXXX. error=N`
CAUSE	Could not find a file. The file is not present or does not have the appropriate permissions set. N is an internal HP-UX error number. Filename given in XXXX, could be: /etc/ftam_resp /etc/ifconfig /etc/mms_spp /etc/osidiag /etc/x25/x25install /etc/x25init /usr/bin/cmistart /usr/bin/cmistat /usr/bin/cmistop /usr/bin/otsstart /usr/bin/otsupdate /usr/bin/sam /usr/lib/osiconf/osiconfx /usr/lib/x400/x4start /usr/lib/x400/x4configprint /usr/lib/x400/x4admin /usr/lib/X400/x4xfer /usr/lib/x500/x5start /usr/lib/x500/x5admin /usr/lib/x500/x5config /usr/lib/x500/x5viewconf Any config file specified in an x25init line in /etc/netlinkrc
ACTION	Make sure the files are present and the permissions are set.

MESSAGE	`Can't find /etc/netlinkrc file.`
CAUSE	/etc/netlinkrc file does not exist, or the permissions are not set.
ACTION	Create the netlinkrc file or set the permissions correctly.

MESSAGE	`grep_field: Can't open XXXX.`
CAUSE	Unable to open the X.25 configuration file XXXX that is specified in /etc/netlinkrc.
ACTION	Correct the X.25 configuration file referenced by the x25init command in /etc/netlinkrc.

MESSAGE	`grep_field: Pattern XXXX not found in YYYY.`
CAUSE	An expected pattern was not found in a file. YYYY will be an X.25 configuration file specified in /etc/netlinkrc. XXXX will be one of: ^ [Xx].121 ^ [\t]*device ^ [\t]*name
ACTION	A line starting with x.121, device, or name, was not found in the specified configuration file. Edit the /etc/netlinkrc file or the configuration file, or X.25 has not been configured correctly.

MESSAGE	`grep_field: There are less than 2 fields.`
CAUSE	X.25 has not been configured correctly. In one of the configuration files, a line starting with x.121, device, or name, does not have any other data on the line.
ACTION	Correct the X.25 configuration file.

MESSAGE	`Can't find /XXXX/ots_subnets file.`
CAUSE	The file is either missing or unreadable. XXXX will usually be the /etc/net/osi/conf directory; however, it may be different if the $OSI_CONFIG environment variable is set.
ACTION	Change the permissions or create the file.

MESSAGE	**get_ots_link_info: No interface name specified for LAN.**
CAUSE	/etc/net/osi/conf/ots_subnets exists but contains erroneous data. A subnet is specified that uses 802.3, but there is no interface name associated with that subnet.
ACTION	Edit /etc/net/osi/conf/ots_subnets to include the interface name.

MESSAGE	**get_ots_link_info: OTS subnets not configured.**
CAUSE	There are no subnets configured in /etc/net/osi/conf/ots_subnets.
ACTION	Run *osiconf* or edit /etc/net/osi/conf/ots_subnets to configure one or more subnets.

MESSAGE	**insert_cmd: Insertion failed.**
CAUSE	OSIADMIN is attempting to write information into /etc/netlinkrc. It is running up against a size limit that prevents it from adding the line.
ACTION	Remove extraneous or unnecessary information from /etc/netlinkrc.

MESSAGE	**Sorry, you must be super user to run OSIADMIN.**
CAUSE	The user trying to invoke *osiadmin* is not the super user.
ACTION	Get the super user password and login.

MESSAGE	**x25viewconf: popen failed. error=N**
CAUSE	An attempt to open a pipe between X.25stat and *osiadmin* failed. The file system may be full, too many file descriptors may be open. N is a Unix internal error code, that may provide additional information on the pipe failure.
ACTION	Remove files that are not needed, to allow for adequate space.

MESSAGE	**`viewconfig: Couldn't open XXXX file.`**
CAUSE	The user is trying to view configuration information that is missing or unreadable. XXXX will usually be the /etc/net/osi/conf directory as shown below; however, it may be different if the $OSI_CONFIG environment variable is set. /etc/net/osi/conf/ots_parms /etc/net/osi/conf/ots_dests /etc/net/osi/conf/ots_routes /etc/net/osi/conf/ots_subnets /etc/net/osi/conf/local_app /etc/net/osi/conf/remote_app /etc/net/osi/conf/mms_parms /etc/net/osi/conf/cm_user_app
ACTION	Make sure the file exists and has the appropriate permissions.

MESSAGE	**`No need to re-start OTS.`**
CAUSE	An attempt to start OTS was made while OTS was already running.
ACTION	None. OTS is already running.

MESSAGE	**`viewconfig: XXXX contains no configurable data.`**
CAUSE	The given file does not have any configuration information in it. XXXX will usually be the /etc/net/osi/conf directory as shown below; however, it may be different if the $OSI_CONFIG environment variable is set. /etc/net/osi/conf/ots_parms /etc/net/osi/conf/ots_dests /etc/net/osi/conf/ots_routes /etc/net/osi/conf/ots_subnets /etc/net/osi/conf/local_app /etc/net/osi/conf/remote_app /etc/net/osi/conf/mms_parms /etc/net/osi/conf/cm_user_app
ACTION	The file must be edited using *osiconf* or a text editor to put configuration information in it.

MESSAGE	**X.25 has not been configured.**
CAUSE	There are no x25init lines in /etc/netlinkrc.
ACTION	Add the appropriate line to /etc/netlinkrc to configure X.25.

MESSAGE	**No device has been specified in XXXX file.**
CAUSE	X.25 has not been configured correctly. In file XXXX, a line starting with device, does not have any other data on the line.
ACTION	The user must fix the configuration file.

MESSAGE	**exec_command: popen failed errno=N**
CAUSE	An attempt to open a pipe between *osiadmin* and another command failed. Filesystem may be full, or too many file descriptors may be open. N is a Unix internal error code that may provide more information on the pipe failure.
ACTION	Remove files that are not needed, to allow for adequate space.

MESSAGE	**LAN has not been configured.**
CAUSE	There are no ifconfig lines in /etc/netlinkrc.
ACTION	Add the appropriate line to /etc/netlinkrc to configure LAN.

MESSAGE	**check_links: No ifconfig entry in netlinkrc.**
CAUSE	There is no ifconfig entry in /etc/netlinkrc even though a LAN link is configured in /etc/net/osi/conf/ots_subnets. (Note that the ots_subnets file may be in a different directory if the $OSI_CONFIG environment variable is set.)
ACTION	The user should add an ifconfig line to /etc/netlinkrc.

MESSAGE	`check_links: No entry for XXXX in netlinkrc`
CAUSE	There is no ifconfig entry in /etc/netlinkrc for the device file associated with the LAN subnet in ots_subnets. XXXX should be the unit name associated with device file of the LAN subnet as specified in /etc/net/osi/conf/ots_subnets. (Note that the ots_subnets file may be in a different directory if the $OSI_CONFIG environment variable is set.)
ACTION	Add the ifconfig to the line in /etc/netlinkrc or change ots_subnets.
MESSAGE	`check_links: No x25init entry in netlinkrc.`
CAUSE	There is no x25init entry in /etc/netlinkrc even though an X25 link is configured in /etc/net/osi/conf/ots_subnets. (Note that the ots_subnets file may be in a different directory if the $OSI_CONFIG environment variable is set.)
ACTION	Add an x25init line to /etc/netlinkrc.
MESSAGE	`check_links: Status of XXXX - not up after 30 seconds.`
CAUSE	An attempt to start the XXXX X.25 card referenced in ots_subnets failed. Card still not up after 30 seconds.
ACTION	Re-configure X.25 card, and refer to X.25 documents for troubleshooting information.
MESSAGE	`check_links: X.25 not configured for XXXX access name.`
CAUSE	There is no x25init entry in /etc/netlinkrc for the programmatic access names configured in /etc/net/osi/conf/ots_subnets. (Note that the ots_subnets file may be in a different directory if the $OSI_CONFIG environment variable is set.) XXXX should be a programmatic access name referenced in ots_subnets.
ACTION	Add the x25init line to /etc/netlinkrc or change /etc/net/osi/conf/ots_subnets.

MESSAGE	`x25conntest: x.121 address is not specified in XXXX.`
CAUSE	The entry for X.121 address is missing in X.25 configuration file. XXXX is the name of the configuration file.
ACTION	Edit the X.25 configuration file and put in the entry for x.121 address.

MESSAGE	`x25conntest: Programmatic access name is not specified in XXXX.`
CAUSE	The entry for programmatic access name is missing in X.25 configuration file. XXXX is the name of the configuration file.
ACTION	Edit the X.25 configuration file and put in the entry for programmatic access name.

MESSAGE	`get_x25_dev_status: Couldn't open device XXXX. errno=N`
CAUSE	Open system call failed to open the device file XXXX.
ACTION	Check for the errno N to determine what went wrong.

MESSAGE	`get_x25_dev_status: ioctl failed. errno=N`
CAUSE	Ioctl system call failed.
ACTION	Check for the errno N to determine what went wrong.

MESSAGE	`x25init command(s) ignored while reading the /etc/netlinkrc file.`
CAUSE	Either the c option was missing, or no config file name was given.
ACTION	Correct or add the required x25init command in the /etc/netlinkrc file.

MESSAGE	`OTS must be running before attempting an update.`
CAUSE	An attempt was made to execute *otsupdate* when OTS was down.
ACTION	To invoke any changes, whether dynamic or not, simply start OTS.

MESSAGE	`Can't create temp file.`
CAUSE	Unable to open the /tmp/OSIXXXXXX file in write mode.
ACTION	Check the /tmp/OSIXXXXXX file permissions.
MESSAGE	`get_lan_addr: popen failed errno = N`
CAUSE	An attempt to open a pipe between *osiadmin* and *lanscan* failed. Filesystem may be full, or too many file descriptors may be open. N is an HP-UX internal error code that may provide more information on the pipe failure.
ACTION	Remove files that are not needed, to allow for adequate space.
MESSAGE	`get_lan_addr: can't get address for XXXX from` *`lanscan`*`.`
CAUSE	The lan station address for XXXX was not found in the *lanscan* output.
ACTION	Check that the given LAN (XXXX) exists using *lanscan* and configure it, if necessary, using *ifconfig*.
MESSAGE	`get_lan_type: popen fail errno = N`
CAUSE	An attempt to open a pipe between *osiadmin* and *lanscan* failed. Filesystem may be full, or too many file descriptors may be open. N is an HP-UX internal error code that may provide more information on the pipe failure.
ACTION	Remove files that are not needed, to allow for adequate space.
MESSAGE	`LAN Interface XXXX: YYYY`
CAUSE	An error, YYYY, occurred while checking the XXXX LAN link.
ACTION	Refer to the message indicated by YYYY in this chapter.
MESSAGE	`X.25 XXXX: YYYY`
CAUSE	An error, YYYY, occurred while checking the XXXX X.25 link.
ACTION	Refer to the message indicated by YYYY in this chapter.

OTSSTART

MESSAGE	**otsstart: OTS directory does not exist: /etc/net/osi/ots**
CAUSE	The /etc/net/osi/ots directory has been (re)moved.
ACTION	Restore this directory, or reinstall the OTS product.
MESSAGE	**otsstart: cannot read OTS file: /etc/net/osi/ots/otspath.env**
CAUSE	The /etc/net/osi/ots/otspath.env file either does not exist or does not have read permission set.
ACTION	Restore this file with read permission or reinstall the OTS product.
MESSAGE	**otsstart: cannot execute OTS program: <file> (<env var>)**
CAUSE	The program indicated by <*EI*> either does not exist or is not executable.
ACTION	Restore the indicated program with execute permission or reinstall the OTS product. The var shown is for HP internal use.
MESSAGE	**OTS already running. You must reboot between startups.**
CAUSE	OTS can be started only once per system boot. This message indicates that OTS is already running.
ACTION	If you need to restart OTS, you must reboot the system first.
MESSAGE	**otsstart: cannot be interrupted**
CAUSE	An attempt was made to interrupt the *otsstart* process. The interrupt was issued too far into the *otsstart* process and interrupting would leave OTS in an unrecoverable state. The interrupt was ignored.

MESSAGE	`otsstart: process interrupted`
CAUSE	The *otsstart* process was interrupted. OTS was not started.
ACTION	You may invoke otsstart again if desired.

MESSAGE	`Usage: otsstart [-c <OTS translated file>])`
CAUSE	An illegal option or parameter was given to otsstart.
ACTION	Check the syntax of the command and try again.

MESSAGE	`osiconfchk reported a validation error: OTS not started.`
	`run "osiconfchk" to get details on the error.`
	`Use osiadmin to correct your configuration.`

CAUSE	An error in the configuration files was detected. OTS was not started. You may get details on the reported error by executing *osiconfchk.*
ACTION	Correct any configuration errors and try again.

MESSAGE	`osiconfchk reported a validation warning: continuing...`
	`run "osiconfchk" to get details on the warning.`
	`Use osiadmin to correct your configuration if necessary.`

CAUSE	A warning was reported during configuration file validation. This error was not serious enough to prevent OTS from starting, but may have adverse effects on product operation.
ACTION	You may get details on the reported warning by executing *osiconfchk.*

MESSAGE	**otstrans reported a translation error: OTS not started. /etc/net/osi/ots/OTS.translog contains the partially translated configuration.**
CAUSE	An error was reported during the configuration file translation. OTS was not started. Normally, this error occurs as the result of a configuration file error/warning that will be reported by *osiconfchk.*
ACTION	Run *osiconfchk*, correct the error/warning condition and try again. If this message persists, contact HP.

MESSAGE	**otstrans reported a translation warning: continuing... See /etc/net/osi/ots/OTS.translog for a description of the warning condition.**
CAUSE	A warning was reported during configuration file translation. This warning was not serious enough to prevent OTS from starting, but may have adverse effects on product operation. Normally, this error occurs as the result of a configuration file error/warning that will be reported by *osiconfchk.*
ACTION	Run *osiconfchk* to get details on the warning condition to determine if you wish to correct the warning. If you fix the warning condition, and this message persists, contact HP.

MESSAGE	**OTS translated configuration file <file> not present: OTS not started.**
CAUSE	The indicated <file> does not exist. Normally, this file is created as a part of the *otsstart* process; however, <file> may be specified by the -c option of *otsstart.* If -c was specified, it is most likely that <file> simply does not exist. If -c was not specified, it is possible that some other (non OTS) process is deleting <file>. Note: the -c option of otsstart is not supported for customer use.

MESSAGE	`OTS not started`
CAUSE	An error was encountered when starting the Streams scheduler, /etc/str_sched.
ACTION	Make sure that this file exists and is executable. If this file was deleted, reinstall OTS. Check the contents of /etc/net/osi/ots/OTS.startlog for possible additional messages.

MESSAGE	`nettl will be unavailable for OTS.`
CAUSE	An error was encountered when starting the OTS *nettl* log daemon. This error is usually encountered when OTS itself cannot be started for some reason.
ACTION	Check the output of *otsstart* for earlier error messages, correct these problems and try again. It is also possible that the log daemon's configuration file, /etc/net/osi/ots/ots_tl, has been deleted or corrupted. If this is the case, OTS is running, but *nettl* will be unavailable for OTS until the problem is corrected and OTS is restarted. Reinstall OTS to recover the ots_tl file.

OTSUPDATE

MESSAGE	**`otsupdate: OTS directory does not exist: /etc/net/osi/ots`**
CAUSE	The /etc/net/osi/ots directory has been (re)moved.
ACTION	Restore this directory, or reinstall the OTS product.
MESSAGE	**`otsupdate: cannot read OTS file: /etc/net/osi/ots/otspath.env`**
CAUSE	The /etc/net/osi/ots/otspath.env file either does not exist or does not have read permission set.
ACTION	Restore this file with read permission or reinstall the OTS product.
MESSAGE	**`otsupdate: cannot execute OTS program: <file> (<env var>)`**
CAUSE	The program indicated by <file> either does not exist or is not executable.
ACTION	Restore the indicated program with execute permission or reinstall the OTS product. The <env var> shown is for HP internal use.
MESSAGE	**`OTS is not running.  Start OTS to get new configuration to take effect.`**
CAUSE	OTS can only be updated when it is already running.
ACTION	Use *otsstart* instead.
MESSAGE	**`otsupdate: cannot be interrupted`**
CAUSE	An attempt was made to interrupt the *otsupdate* process. The interrupt was issued too far into the *otsupdate* process and interrupting would leave OTS in an unrecoverable state. The interrupt was ignored.

MESSAGE	**`otsupdate: process interrupted`**
CAUSE	The *otsupdate* process was interrupted. OTS was not updated.
ACTION	You may invoke *otsupdate* again if desired.
MESSAGE	**`Usage: otsupdate [-fm | -c <operator update script>]`**
CAUSE	An illegal option or parameter was given to *otsupdate*.
ACTION	Check the syntax of the command and try again.
MESSAGE	**`otsupdate: The -f and -c options cannot both be specified.`**
CAUSE	The -f and -c options are conflicting.
ACTION	See the *otsupdate* man page for details on the use of these options.
MESSAGE	**`otsupdate: The -m and -c options cannot both be specified.`**
CAUSE	The -m and -c options are conflicting.
ACTION	See the *otsupdate* man page for details on the use of these options.

MESSAGE	**`osiconfchk reported a validation error: OTS not updated.`**
	`run "osiconfchk" to get details on the error.`
	`Use osiadmin to correct your configuration.`

CAUSE	An error in the configuration files was detected. OTS was not updated. You may get details on the reported error by executing *osiconfchk*.
ACTION	Correct any configuration errors and try again.

MESSAGE	**osiconfchk reported a validation warning: continuing... run "osiconfchk" to get details on the warning. Use osiadmin to correct your configuration if necessary.**
CAUSE	A warning was reported during configuration file validation. This error was not serious enough to prevent OTS from updating, but may have adverse effects on product operation.
ACTION	You may get details on the reported warning by executing *osiconfchk.*

MESSAGE	**osiconfchk reported a non-dynamic change: OTS not updated. run "osiconfchk -u" to get details on the error. To use new configuration, reboot the system and restart OTS.**
CAUSE	A change to the configuration files was made that cannot be incorporated into the OTS product without rebooting the system. OTS was not updated. Running *osiconfchk -u* will tell you the file in which the non-dynamic change was made, but will not tell you what specific change caused the error.
ACTION	If you can fix the error, you may try *otsupdate* again, otherwise, you will have to reboot the system and restart OTS in order to get your changes to take effect.

MESSAGE	**otstrans reported a translation error: OTS not updated. /etc/net/osi/ots/OTS.translog contains the partially translated configuration.**
CAUSE	An error was reported during the configuration file translation. OTS was not updated. Normally, this error occurs as the result of a configuration file error/warning that will be reported by *osiconfchk.*
ACTION	Run *osiconfchk*, correct the error/warning condition and try again. If this message persists, contact HP.

MESSAGE	**`otstrans reported a translation warning: continuing... See /etc/net/osi/ots/OTS.translog for a description of the warning condition.`**
CAUSE	A warning was reported during configuration file translation. This warning was not serious enough to prevent OTS from updating, but may have adverse effects on product operation. Normally, this error occurs as the result of a configuration file error/warning that will be reported by *osiconfchk*.
ACTION	Run *osiconfchk* to get details on the warning condition to determine if you wish to correct the warning. If you fix the warning condition, and this message persists, contact HP.

MESSAGE	**`OTS translated configuration file <file> not present: OTS not updated.`**
CAUSE	The indicated <file> does not exist. Normally, this file is created as a part of the *otsupdate* process; however, <file> may be specified by the -c option of *otsupdate*. If -c was specified, it is most likely that <file> simply does not exist. If -c was not specified, it is possible that some other (non OTS) process is deleting <file>.
	Note: the -c option of otsstart is not supported for customer use.

Protocol Reason Codes

This section describes the reason codes which may be logged or returned to your program in the following situations.

- Session Provider Abort Reason Code - The value passed back to Session programs by the ses_pabort_id() routine.
- Transport Disconnect Reason Code - The value passed back to XTI programs by the t_rcvdis() routine.
- ACSE/Presentation DCNX_KO Log Message - The value shown in the second low order byte of the Cause field (that is, the "f3" in Cause = 0x0001f3ff).
- Session S_REJECT Log Message - The value shown in the second low order byte of the Cause field (that is, the "f3" in Cause = 0x0001f3ff).
- Transport T_REJECT Log Message - The value shown in the second low order byte of the Cause field (that is, the "f3" in Cause = 0x0001f3ff).
- Network N_REJECT Log Message - The value shown in the low order byte of the Org/Reas field (that is, the "08" in Org/Reas=0108).

The following list gives the reason code value, its meaning, and possible corrective actions.

- **(0) Normal Disconnect.** - The address you have specified may be correct, but there is no process listening for a connection. Verify that any OSI services are up on the remote. Also verify the T-selector specified.
- **(0x01) Provider N-Disconnect. (Transient)** - A previously active Network connection was abruptly disconnected. This may indicate that the X.25 card or the switch went down; OR

  (0x01) Congestion at TSAP - On some vendor's equipment this may indicate that the application above transport is not capable of receiving any more connections. Check the remote for any further logged information.
- **(0x02) Provider N-Disconnect. (Permanent)** - A previously active Network connection was abruptly disconnected. This may indicate that the X.25 card or the switch went down; OR

  (0x02) Transport user not attached to TSAP - This indicates that the address specified is valid, but that no process capable of receiving connections is active. Verify that a server is active on the remote.
- **(0x03) Provider N-Reject. (Transient)** - The Network connection could not be established; OR

(0x03) Address Unknown - The address specified was incorrect. Verify both the NSAP and the T-selector.

- **(0x04) Provider N-Reject. (Permanent)** - Our request to establish a Network Connection was rejected. Possible errors might be facility rejection, non-use of fast select, reverse charge failure, switch or PDN out of order.
- **(0x05) Provider N-Reject. (QOS unavail/Transient)** - QOS negotiation failed for this connection.
- **(0x06) Provider N-Reject. (QOS unavail/Permanent)** - QOS negotiation failed for this connection.
- **(0x07) N-Reject. (NSAP unreachable / Transient)** - Our Network connection could not be established because there were not enough VCs or resources available. Use *x25stat* to examine the state of the network card. Also check the log file for OTS messages.
- **(0x08) N-Reject. (NSAP unreachable / Permanent)** - Our Network connection could not be established, because the X.121 address was incorrect. Use *osiadmin* or *otsshowes* to examine the configured X.121 address for this NSAP. Also verify remote is actually up and listening on the address configured.
- **(0x09) N-Reset from NS provider. (No Reason)** - The network connection was reset. No recognized diagnostic was provided.
- **(0x0a) N-Reset from NS provider. (Congestion)** - The network connection was reset. The X.25 diagnostic indicates provider congestion was a problem.
- **(0x0b) N-Reject. (Address Unknown)** - No destination or route entry exists for the given NSAP. As a result the Network connection can not be established. Use *otsadmin* or *otsaddes* to configure the destination system.
- **(0x12) Disconnect from NS user.** - The remote Transport Layer encountered a failure and abruptly released the Network connection.
- **(0x14) User N-Reject. (Transient)** - The remote Transport Layer refused our Network connection request.
- **(0x15) User N-Reject. (Permanent)** - The remote Transport Layer refused our Network connection request.
- **(0x16) User N-Reject. (QOS unavail/Transient)** - QOS negotiation failure.
- **(0x17) User N-Reject. (QOS unavail/Permanent)** - QOS negotiation failure.
- **(0x18) N-Reject. (Unrecognized NS-user-data)**
- **(0x1a) Network Reset from NS user. (Resynch)** - The network connection was reset. The X.25 diagnostic indicates resynchronization was requested.

- **(0x20) Undefined reason, unknown origin.** - This indicates some X.25 failure. Possibilities include the card going down at either the local or remote. Switch problems. Or facilities negotiation (for example, reverse charging).
- **(0x21) Invalid parameter. (OTS Kernel)** - A parameter we were encoding was invalid. Typically this is with NSAPs (too large or absent).
- **(0x22) External protocol error. (OTS Kernel)** - Failure decoding facility information. An error should be logged describing the protocol error encountered.
- **(0x23) Invalid internal state. (OTS Kernel)** - A network primitive was received or is to be sent in an unexpected state (that is, N-Connect.Cf after connected). An error should be logged with more information.
- **(0x24) Facility or data field overflow. (OTS Kernel)** - Processing of the facility fields indicated failure. An error should be logged with more information.
- **(0x41) X.25 facility requested not allowed.** - The configured X.25 facilities do not match those of the provider. Determine the facilities available from the provider by contacting the X.25 Network Administrator and reconfigure OTS and X.25 to use the appropriate set of facilities.
- **(0x42) Could not access X.121 address of remote.** - The X.121 address mapped to the NSAP you specified is incorrect or the remote system is down. Reset the remote system or reconfigure the OTS routing table appropriately.
- **(0x81) Remote transport entity congestion at connect time** - This indicates that no problems were detected with the Transport layer, but the user of the service may have previously sent an abort or other higher level error.
- **(0x82) Connection negotiation failed** - This may be a problem with the classes specified. The remote may only use class 0. If this is the case you might reconfigure the local stack to force class 0, this is a Transport over CONS parameter.
- **(0x83) Duplicate source reference for same NSAP** - This is a protocol error. The caller must generate a unique reference number for each connection it forms. See "Submitting Problem Information to HP" in chapter 4.
- **(0x84) Mismatched references** - This is either protocol error, or may indicate that the remote stack was taken down and then brought back up while existing connections remained. Check the state of the remote stack and see "Submitting Problem Information to HP" in chapter 4 if remote stack indicated no errors.
- **(0x85) Protocol error** - Check for any logged information on the remote side. Also verify the stack was not being brought up or taken down at the time. If problem persists see "Submitting Problem Information to HP" in chapter 4.
- **(0x88) Connection request refused on this network connection** - The level of multiplexing configured for the local node may not be compatible with that on the remote. On HP systems the Transport over CONS parameters

`tpcons_max_con_mux_in` and `tpcons_max_con_mux_out` should be examined and changed if max out exceeds max in.

- **(0x8A) Header or parameter length invalid** - This indicates a protocol error. Check any logged information on the remote. If problem persists see "Submitting Problem Information to HP" in chapter 4.
- **(0xE4) X.25 network error or network down.** - Run X.25 status to determine the status of the X.25 link. You may need to restart X.25 or reset other X.25 hardware (for example, switches).
- **(0xE7) Problem accessing X.25 subaddress of remote.** - The subaddress portion of the X.121 address which is configured in the OTS routing table is incorrect or the remote stack is not up. Correct the routing table entry or bring up the remote as appropriate.
- **(0xE8) NSAP not configured.** - The Network Address portion of the address specified is not configured in the routing table. This problem should not occur if you are trying to go over LAN. Follow the instructions for configuring remote X.25 routes through *osiadmin.*
- **(0xF0) Protocol error detected by remote's Transport.** - The remote encountered an error decoding one of the Transport PDU's sent by us. Generate a trace and contact your HP support representative.
- **(0xF1) Protocol error detected by remote's Session.** - The remote encountered an error decoding one of the Session PDU's sent by us.
- **(0xF3) Invalid address or permanent Transport error.** - This error is typically returned when an invalid T-selector or Network address is given. You may also receive this error if the remote stack is not running. Or if you have not associated the remote's X.121 address with its NSAP through *osiconf.*
- **(0xF4) Invalid address or permanent Session error.** - Check the address specified. Also check the state of the remote stack ensuring all layers are up. Correct the address or remote stack and rerun.
- **(0xF6) X.25 unavailable or transient Transport error.** - Verify that X.25 is up by running the X25 status operation.
- **(0xF7) Transient Session error.** - Verify status of remote stack. If error persists contact your HP support representative.
- **(0xF9) Protocol error detected by local Transport.** - An error was encountered decoding a PDU sent by the remote Transport entity. Generate a trace and contact your HP support representative.
- **(0xFA) Protocol error detected by local Session.** - An error was encountered decoding a PDU sent by the remote Session entity. Generate a trace and contact your HP support representative.

- **(0xFC) Insufficient resources for local Transport.** - The stack is experiencing resource problems, examine other OTS processes running. Contact your HP support representative if condition persists.
- **(0xFD) Insufficient resources for local Session.** - The stack is experiencing resource problems, examine other OTS processes running. Contact your HP support representative if condition persists.
- **(0xFE) Insufficient resources for local ACSE/Presentation.** - The stack is experiencing resource problems; examine other processes using OTS. Are you near the 448 connection limit? Contact your HP support representative if problem persists.
- **(0xFF) Local user error.** - Check parameters whose default values you have overridden. This may result from trying to run an operation (for example, Session Give Token) in the wrong state. If problem persists contact your HP support representative.
- **Code unknown. (01-EF X.25; F0-FF Transport).** - An unrecognized error code was received. Examine the documentation of the other vendor for Disconnect reason codes used for further action. Also if the code is in the X.25 range, then run an X.25 connection test. The diagnostic information provided if this low level test fails may be helpful.

Session Refuse Codes

The following list describes the meanings defined by ISO 8327 for the reason codes carried on a negative Session connect confirmation. These values are logged after S-CONNECT-Resp (Negative) message, and on return from the ses_connect_rf() library call.

0	Reason not specified
1	Rejection by called Session service user due to temporary congestion
2	Rejection by called Session service user. The following octets may be used for up to 512 octets of user data.
81	SSAP identifier unknown
82	Session service user not attached to SSAP
83	SBM congestion at connect time
84	Proposed protocol versions not supported

OTS/9000 Log and Trace Messages

These messages are written to `/usr/adm/nettl.LOGxx` in binary format. (Use *netfmt* to convert the messages into a readable form.) If the error condition continues or poses product or application usage problems, gather the application and OTS/9000 log files, note the operating environment, and contact HP support.

Note Messages whose ACTION section contain "None" are informative and recoverable.

The following shows the general format of an OTS log or trace message.

`[EENN]` *Optional Text < source filename >< source line#>*

Optional Text

where EE = Entity Number and NN = Message ID Number

The optional text may contain: Description, Parameters, and Miscellaneous Information.

Entity Numbers

Entity Title	Entity Number (EE)
Network	30
Transport/CONS	40
Transport/CLNS	41
Session	50
ACSE/Presentation	60
High Access	96
LAN	97
X.25	98
Kernel	99

The error messages listed below do not have further discussion in this section:

3014
4014,4015
4114,4115
5011,5012
6011
9600
9700
9800
9951

The second line of information logged for the above errors will describe the specific problem encountered, such as a protocol error or other anomaly. As an example:

```
[4115] Processing Error trsmai2.c 948
[....] (3) pcc() TPDU size negotiation
```

The first two fields on the second line, "(3)" and "pcc()" are intended for use by HP personnel. The descriptive text, in this case "TPDU size negotiation" indicates what went wrong.

If you cannot isolate this problem, follow the directions in "Submitting Problem Information to HP" section in chapter 4, "Troubleshooting."

MESSAGE	`[4001] T_CTX file line #`
	`[....] ENT=p1 CTX=p2 CHN=p3 SAP=p4 SUF=p5 SUF=p6 SAP=p7 ADD=p8 ADD=p9 CTX=p10 CHN=p11 PDS=p12 REF=p13 REF=p14 Status=p15 NBR=pr1 pr2 pr3 pr4 Fu=p16 Fd=p17`

CAUSE	This is the output from an otsop show or find command. It outputs the current Transport context status.

PARAMETERS

p1	: entity number
p2	: Transport context number
p3	: Transport channel number
p4	: Transport sap number
p5	: local Transport selector
p6	: remote Transport selector
p7	: NSAP
p8	: local network address
p9	: remote network address
p10	: network context number
p11	: network channel number
p12	: PDU size
p13	: local Transport reference
p14	: remote Transport reference
p15	: Hexadecimal bitstring:

0E00 ssss cccc oooo Rccc PPPP PPPP PPPP

E : 1 = connection established

cccc	: Transport class
oooo	: protocol options
ssss	: state at time of message generation

R : 1 = incoming connection

PPPP	: priority
pr1	: next PDU to send
pr2	: local upper window limit
pr3	: next expected PDU
pr4	: remote upper window limit
p16	: flow control upper limit
p17	: flow control lower limit

ACTION None.

MESSAGE	`[4002] N_CTX file line #`
	`[....] TRS=p1 MPX=p2 OWN=p3 Fu=p4 Fd=p5 SAP=p6 ADD=p7`	

CAUSE	This is the output from an otsop show or find command. It outputs the current network context status.	
PARAMETERS	p1	: context number of first associated Transport class
	p2	: number of associated Transport classes
	p3	: 1 = initiator, 0 = responder
	p4	: flow control upper limit
	p5	: flow control lower limit
	p6	: NSAP
	p7	: remote network address
ACTION	None.	

MESSAGE	`[4003] Protocol Error file line #`
	`[....] ENT=p1 NET=p2 TRS=p3 p4 PDU=p5`	

CAUSE	A protocol error has been deleted in the Transport or lower layers.	
PARAMETERS	p1	: Transport entity number
	p2	: network sap number
	p3	: Transport context number
	p4	: Transport context state
	p5	: offending PDU
ACTION	Examine the log file and the offending PDU to determine error cause and the corrective actions.	

MESSAGE	`[4004] User Error file line #`
	`[....] ENT=p1 EVN=p2 TRS=p3 p4`	

CAUSE	A user has attempted an improper action.	
PARAMETERS	p1	: Transport entity number
	p2	: interface event
	p3	: Transport context number
	p4	: Transport connection state
ACTION	Examine the connection state and error cause to determine corrective actions.	

MESSAGE	**[4005] Retrans Limit file line #** **[....] ENT=p1 NET=p2 TRS=p3 p4 PDU=p5**
CAUSE	The number of times that the Transport will attempt to transmit a PDU has been exceeded.
PARAMETERS	p1 : Transport entity number p2 : network sap number p3 : Transport context number p4 : Transport context state
ACTION	Use network management tools available to determine if the remote system is still reachable. If the node is reachable, examine network congestion and quality for possible delays or lost data. The OTS/9000 product uses slow-start and congestion management routines to minimize reduced service quality effects, but it is still possible to loose PDUs or connections.

MESSAGE	**[4006] Inactivity Timer file line #** **[....] ENT=p1 NET=p2 TRS=p3 p4 PDU=p5**
CAUSE	The local Transport provider has lost contact with the peer Transport Layer and has aborted the connection.
PARAMETERS	p1 : Transport entity number p2 : network sap number p3 : Transport context number p4 : Transport connection state
ACTION	Examine the log file and the offending PDU to determine the cause. It is possible that nothing can be done from the operator's end, for example, the remote sends a bogus request and the stack aborts the connection. Examine the following output to determine the cause and any corrective actions to take.

MESSAGE	`[4007] T_ESTAB file line #`
	`[....] ENT=p1 CTX=p2 CHN=p3 SAP=p4 SUF=p5 SUF=p6 SAP=p7 ADD=p8 ADD=p9 CTX=p10 CHN=p11 PDS=p12 REF=p13 REF=p14 Status=p15`

CAUSE	Transport connection establishment is being attempted between two endpoints.

PARAMETERS

p1	: entity number
p2	: Transport context number
p3	: Transport channel number
p4	: Transport sap number
p5	: local Transport selector
p6	: remote Transport selector
p7	: NSAP
p8	: local network address
p9	: remote network address
p10	: network context number
p11	: network channel number
p12	: PDU size
p13	: local Transport reference
p14	: remote Transport reference
p15	: Hexadecimal bitstring:

0E00 ssss cccc oooo Rccc PPPP PPPP PPPP

E	: 1 = connection established
cccc	: Transport class
oooo	: protocol options
ssss	: state at time of message generation
R	: 1 = incoming connection
PPPP	: priority

ACTION None.

MESSAGE	`[4008] T_REJECT file line #`
	`[....] ENT=p1 CTX=p2 CHN=p3 SAP=p4 SUF=p5 SUF=p6 SAP=p7 ADD=p8 ADD=p9 CTX=p10 CHN=p11 PDS=p12 REF=p13 REF=p14 Status=p15 Cause=p16`

CAUSE	A Transport connect request has been rejected.

PARAMETERS

p1	: entity number
p2	: Transport context number
p3	: Transport channel number
p4	: Transport sap number
p5	: local Transport selector
p6	: remote Transport selector
p7	: NSAP
p8	: local network address
p9	: remote network address
p10	: network context number
p11	: network channel number
p12	: PDU size
p13	: local Transport reference
p14	: remote Transport reference
p15	: Hexadecimal bitstring:

0E00 ssss cccc oooo Rccc PPPP PPPP PPPP

E	: 1 = connection established
cccc	: Transport class
oooo	: protocol options
ssss	: state at time of message generation
R	: 1 = incoming connection
PPPP	: priority

p16 : Hexadecimal bitstring:

0000 0000 0000 000r uuuu uuuu pppp pppp

r	: 1 = disconnect origin is remote
uuuu	: reason delivered to user
pppp	: reason used in protocol

ACTION Examine the Cause parameter to determine the reason and the corrective actions.

MESSAGE	`[4009] T_DCNX_OK file line #`
	`[....] ENT=p1 CTX=p2 CHN=p3 SAP=p4 SUF=p5 SUF=p6 SAP=p7 ADD=p8 ADD=p9 CTX=p10 CHN=p11 PDS=p12 REF=p13 REF=p14 Status=p15`

CAUSE	A normal Transport connection disconnect has occurred.

PARAMETERS

p1	: entity number
p2	: Transport context number
p3	: Transport channel number
p4	: Transport sap number
p5	: local Transport selector
p6	: remote Transport selector
p7	: NSAP
p8	: local network address
p9	: remote network address
p10	: network context number
p11	: network channel number
p12	: PDU size
p13	: local Transport reference
p14	: remote Transport reference
p15	: Hexadecimal bitstring:

0E00 ssss cccc oooo Rccc PPPP PPPP PPPP

E	: 1 = connection established
cccc	: Transport class
oooo	: protocol options
ssss	: state at time of message generation
R	: 1 = incoming connection
PPPP	: priority

ACTION None.

MESSAGE	`[4010] T_DCNX_KO file line #`
	`[....] ENT=p1 CTX=p2 CHN=p3 SAP=p4 SUF=p5 SUF=p6 SAP=p7 ADD=p8 ADD=p9 CTX=p10 CHN=p11 PDS=p12 REF=p13 REF=p14 Status=p15 Cause=p16`

CAUSE	An abnormal Transport connection disconnect has occurred.

PARAMETERS

p1	: entity number
p2	: Transport context number
p3	: Transport channel number
p4	: Transport sap number
p5	: local Transport selector
p6	: remote Transport selector
p7	: NSAP
p8	: local network address
p9	: remote network address
p10	: network context number
p11	: network channel number
p12	: PDU size
p13	: local Transport reference
p14	: remote Transport reference
p15	: Hexadecimal bitstring:

0E00 ssss cccc oooo Rccc PPPP PPPP PPPP

E	: 1 = connection established
cccc	: Transport class
oooo	: protocol options
ssss	: state at time of message generation
R	: 1 = incoming connection
PPPP	: priority

p16 : Hexadecimal bitstring:

0000 0000 0000 000r uuuu uuuu pppp pppp

r	: 1 = disconnect origin is remote
uuuu	: reason delivered to user
pppp	: reason used in protocol

ACTION Examine the status and cause parameters to determine the reason and corrective actions.

MESSAGE	`[4011] T_NET_ASSIGN file line #`
	`[....] ENT=p1 CTX=p2 CHN=p3 SAP=p4 SUF=p5 SUF=p6 SAP=p7 ADD=p8 ADD=p9 CTX=p10 CHN=p11`

CAUSE	The underlying network connection has been reassigned. The message displays the new status of this connection. It is not unusual for connections to be reassigned during a connection's lifetime. Reassignment takes place at layers below the Transport.

PARAMETERS

p1	: entity number
p2	: Transport context number
p3	: Transport channel number
p4	: Transport sap number
p5	: local Transport selector
p6	: remote Transport selector
p7	: NSAP
p8	: local network address
p9	: remote network address
p10	: network context number
p11	: network channel number

ACTION None.

MESSAGE	`[4101] T_CTX file line #`
	`[....] ENT=p1 CTX=p2 CHN=p3 SAP=p4 SUF=p5 SUF=p6 SAP=p7 ADD=p8 ADD=p9 CTX=p10 CHN=p11 PDS=p12 REF=p13 REF=p14 Status=p15 NBR=pr1 pr2 pr3 pr4 Fu=pr5 Fd=pr6`

CAUSE	This is the output from an otsop show or find command. It outputs the current Transport context status.

PARAMETERS

p1	: entity number
p2	: Transport context number
p3	: Transport channel number
p4	: Transport sap number
p5	: local Transport selector
p6	: remote Transport selector
p7	: NSAP
p8	: local network address
p9	: remote network address
p10	: network context number
p11	: network channel number
p12	: PDU size
p13	: local Transport reference
p14	: remote Transport reference
p15	: Hexadecimal bitstring:

0E00 ssss cccc oooo Rccc PPPP PPPP PPPP

E	: 1 = connection established
cccc	: Transport class
oooo	: protocol options
ssss	: state at time of message generation
R	: 1 = incoming connection
PPPP	: priority

pr1	: next PDU to send
pr2	: local upper window limit
pr3	: next expected PDU
pr4	: remote upper window limit
pr5	: flow control upper limit
pr6	: flow control lower limit

ACTION None.

MESSAGE	**`[4103] Protocol Error file line #`**
	`[....] ENT=p1 NET=p2 TRS=p3 p4 PDU=p5`

CAUSE	A protocol error has been detected in the Transport or lower layers.
PARAMETERS	p1 : Transport entity number p2 : network sap number p3 : Transport context number p4 : Transport context state p5 : offending PDU
ACTION	Examine the log file and the offending PDU to determine error cause and the corrective actions.

MESSAGE	**`[4104] User Error file line #`**
	`[....] ENT=p1 EVN=p2 TRS=p3 p4`

CAUSE	A user has attempted an improper action and a user error has been noted.
PARAMETERS	p1 : Transport entity number p2 : interface event p3 : Transport context number p4 : Transport connection state
ACTION	Examine the connection state and error cause to determine corrective actions.

MESSAGE	**[4105] Retrans Limit file line #** **[....] ENT=p1 NET=p2 TRS=p3 p4 PDU=p5**
CAUSE	The number of times that the Transport will attempt to retransmit a PDU has been exceeded. p1 : Transport entity number p2 : network sap number p3 : Transport context number p4 : Transport context state
ACTION	Use network management tools available to determine if the remote system is still reachable. If the node is reachable, examine network congestion and quality for possible delays or lost data. The OTS/9000 product uses slow-start and congestion management routines to minimize reduced service quality effects, but it is still possible to loose PDUs or connections.

MESSAGE	**[4106] Inactivity Timer file line #** **[....] ENT=p1 NET=p2 TRS=p3 p4 PDU=p5**
CAUSE	The local Transport provider has lost contact with the peer Transport layer and has aborted the connection.
PARAMETERS	p1 : Transport entity number p2 : network sap number p3 : Transport context number p4 : Transport connection state
ACTION	Examine the log file and the offending PDU to determine the cause. It is possible that nothing can be done from the operator's end, for example, the remote sends a bogus request and the stack aborts the connection. Examine the following output to determine the cause and any corrective actions to take.

MESSAGE	**[4107] T_ESTAB file line #** **[....] ENT=p1 CTX=p2 CHN=p3 SAP=p4 SUF=p5 SUF=p6 SAP=p7 ADD=p8 ADD=p9 CTX=p10 CHN=p11 PDS=p12 REF=p13 REF=p14 Status=p15**
CAUSE	Transport connection establishment is being attempted between two endpoints.

PARAMETERS

p1	: entity number
p2	: Transport context number
p3	: Transport channel number
p4	: Transport sap number
p5	: local Transport selector
p6	: remote Transport selector
p7	: NSAP
p8	: local network address
p9	: remote network address
p10	: network context number
p11	: network channel number
p12	: PDU size
p13	: local Transport reference
p14	: remote Transport reference
p15	: Hexadecimal bitstring:

0E00 ssss cccc oooo Rccc PPPP PPPP PPPP

E	: 1 = connection established
cccc	: Transport class
oooo	: protocol options
ssss	: state at time of message generation
R	: 1 = incoming connection
PPPP	: priority

ACTION None.

MESSAGE	**[4108] T_REJECT file line #** **[....] ENT=p1 CTX=p2 CHN=p3 SAP=p4 SUF=p5 SUF=p6 SAP=p7 ADD=p8 ADD=p9 CTX=p10 CHN=p11 PDS=p12 REF=p13 REF=p14 Status=p15 Cause=p16**
CAUSE	A Transport connect request has been rejected.

PARAMETERS

p1	: entity number
p2	: Transport context number
p3	: Transport channel number
p4	: Transport sap number
p5	: local Transport selector
p6	: remote Transport selector
p7	: NSAP
p8	: local network address
p9	: remote network address
p10	: network context number
p11	: network channel number
p12	: PDU size
p13	: local Transport reference
p14	: remote Transport reference
p15	: Hexadecimal bitstring:

0E00 ssss cccc oooo Rccc PPPP PPPP PPPP

E	: 1 = connection established
cccc	: Transport class
oooo	: protocol options
ssss	: state at time of message generation
R	: 1 = incoming connection
PPPP	: priority

p16 : Hexadecimal bitstring:

0000 0000 0000 000r uuuu uuuu pppp pppp

r	: 1 = disconnect origin is remote
uuuu	: reason delivered to user
pppp	: reason used in protocol

ACTION Examine the Cause parameter to determine the reason and the corrective actions.

MESSAGE	**[4109] T_DCNX_OK file line #** **[....] ENT=p1 CTX=p2 CHN=p3 SAP=p4 SUF=p5 SUF=p6 SAP=p7 ADD=p8 ADD=p9 CTX=p10 CHN=p11 PDS=p12 REF=p13 REF=p14 Status=p15**
CAUSE	A normal Transport connection disconnect has occurred.

PARAMETERS

p1	: entity number
p2	: Transport context number
p3	: Transport channel number
p4	: Transport sap number
p5	: local Transport selector
p6	: remote Transport selector
p7	: NSAP
p8	: local network address
p9	: remote network address
p10	: network context number
p11	: network channel number
p12	: PDU size
p13	: local Transport reference
p14	: remote Transport reference
p15	: Hexadecimal bitstring:

0E00 ssss cccc oooo Rccc PPPP PPPP PPPP

E	: 1 = connection established
cccc	: Transport class
oooo	: protocol options
ssss	: state at time of message generation
R	: 1 = incoming connection
PPPP	: priority

ACTION None.

MESSAGE	`[4110] T_DCNX_KO file line #`
	`[....] ENT=p1 CTX=p2 CHN=p3 SAP=p4 SUF=p5 SUF=p6 SAP=p7 ADD=p8 ADD=p9 CTX=p10 CHN=p11 PDS=p12 REF=p13 REF=p14 Status=p15 Cause=p16`

CAUSE	An abnormal Transport connection disconnect has occurred.

PARAMETERS

p1	: entity number
p2	: Transport context number
p3	: Transport channel number
p4	: Transport sap number
p5	: local Transport selector
p6	: remote Transport selector
p7	: NSAP
p8	: local network address
p9	: remote network address
p10	: network context number
p11	: network channel number
p12	: PDU size
p13	: local Transport reference
p14	: remote Transport reference
p15	: Hexadecimal bitstring:

0E00 ssss cccc oooo Rccc PPPP PPPP PPPP

E	: 1 = connection established
cccc	: Transport class
oooo	: protocol options
ssss	: state at time of message generation
R	: 1 = incoming connection
PPPP	: priority

p16 : Hexadecimal bitstring:

0000 0000 0000 000r uuuu uuuu pppp pppp

r	: 1 = disconnect origin is remote
uuuu	: reason delivered to user
pppp	: reason used in protocol

ACTION Examine the status and cause parameters to determine the reason and corrective actions.

MESSAGE	`[4112] CLNP lf-quantum = p1`
CAUSE	Life time time value. This is the result of an operator command.
PARAMETERS	p1 : time values in number of 500 ms.
ACTION	None.

MESSAGE

```
[5001] S_CTX file line #
[....] ENT=p1 CTX=p2 CHN=p3 SAP=p4 SUF=p5 SUF=p6
SAP=p7 ADD=p8 CHN=p9 Status=p10
```

CAUSE

This is the output from the otsop show or find command. It shows the current context status.

PARAMETERS

p1	: entity number
p2	: Session context number
p3	: Session channel number
p4	: Session sap number
p5	: local Session selector
p6	: remote Session selector
p7	: Transport sap number
p8	: remote Transport selector
p9	: Transport channel number
p10	: hexadecimal bitstring:

0ER0 ssss 0000 0TVV FFFF FFFF FFFF FFFF

E	: 1 = connection established
R	: 1 = incoming connection
T	: 1 = Texp in use
VV	: Session version number 01=1, 10=2, or 11=either, based on negotiation
ssss	: state at time of message generation
FFFF	: functional units 2 bytes

ACTION

None.

MESSAGE	`[5003] Protocol Error file line #`
	`[....] ENT=p1 SES=p2 ssss PDU=p3`	

CAUSE	A protocol error has been detected in the Session layer or below.	
PARAMETERS	p1	: Session entity number
	p2	: Session context number
	ssss	: Session context state
	p3	: offending PDU in binary
ACTION	Check the log file and the offending PDU to determine the cause and corrective action.	

MESSAGE	`[5004] User Error file line #`
	`[....] ENT=p1 EVN=p2 SES=p3 ssss`	

CAUSE	A user has attempted an improper action.	
PARAMETERS	p1	: Session entity number
	p2	: interface event
	p3	: Session context number
	ssss	: Session context state
ACTION	Examine the connection state and error cause to determine corrective actions.	

MESSAGE	`[5007] S_ESTAB file line #`
	`[....] ENT=p1 CTX=p2 CHN=p3 SAP=p4 SUF=p5 SUF=p6 SAP=p7 ADD=p8 CHN=p9 Status=p10`

CAUSE	Connection establishment is being attempted between two endpoints.
PARAMETERS	

p1	: entity number
p2	: Session context number
p3	: Session channel number
p4	: Session sap number
p5	: local Session selector
p6	: remote Session selector
p7	: Transport sap number
p8	: remote Transport selector
p9	: Transport channel number
p10	: hexadecimal bitstring:

0ER0 ssss 0000 0TVV FFFF FFFF FFFF FFFF

E	: 1 = connection established
R	: 1 = incoming connection
T	: 1 = Texp in use
VV	: Session version number: 01=1, 10=2, or 11=either, based on negotiation
ssss	: state at time of message generation

FFFF: functional units 2 bytes

ACTION None.

MESSAGE	**[5008] S_REJECT file line #** **[....] ENT=p1 CTX=p2 CHN=p3 SAP=p4 SUF=p5 SUF=p6 SAP=p7 ADD=p8 CHN=p9 Status=p10 Cause=p11**
CAUSE	A Session connect request has been rejected.

PARAMETERS

p1	: entity number
p2	: Session context number
p3	: Session channel number
p4	: Session sap number
p5	: local Session selector
p6	: remote Session selector
p7	: Transport sap number
p8	: remote Transport selector
p9	: Transport channel number
p10	: Hexadecimal bitstring:

0ER0 ssss 0000 0TVV FFFF FFFF FFFF FFFF

E	: 1 = connection established
R	: 1 = incoming connection
T	: 1 = Texp in use
VV	: Session version number: 01=1, 10=2, or 11=both
ssss	: state at time of message generation
FFFF	: functional units 2 bytes

p11 : Hexadecimal bitstring:

0000 0000 0000 000r uuuu uuuu pppp pppp

r	: 1 = disconnect origin is remote
uuuu	: reason delivered to user
pppp	: reason used in protocol

ACTION Examine the Cause parameter to determine the reason and the corrective actions.

MESSAGE	`[5009] S_DCNX_OK file line #`
	`[....] ENT=p1 CTX=p2 CHN=p3 SAP=p4 SUF=p5 SUF=p6`
	`SAP=p7 ADD=p8 CHN=p9 Status=p10`

CAUSE	A normal Session connection disconnect has occurred.
PARAMETERS	p1 : entity number p2 : Session context number p3 : Session channel number p4 : Session sap number p5 : local Session selector p6 : remote Session selector p7 : Transport sap number p8 : remote Transport selector p9 : Transport channel number p10 : Hexadecimal bitstring: 0ER0 ssss 0000 0TVV FFFF FFFF FFFF FFFF E : 1 = connection established R : 1 = incoming connection T : 1 = Texp in use VV : Session version number: 01=1, 10=2, or 11=both ssss : state at time of message generation FFFF : functional units 2 bytes
ACTION	None.

MESSAGE	`[5010] S_DCNX_KO file line #`
	`[....] ENT=p1 CTX=p2 CHN=p3 SAP=p4 SUF=p5 SUF=p6 SAP=p7 ADD=p8 CHN=p9 Status=p10 Cause=p11`

CAUSE	An abnormal Session connection disconnect has occurred.

PARAMETERS

p1	: entity number
p2	: Session context number
p3	: Session channel number
p4	: Session sap number
p5	: local Session selector
p6	: remote Session selector
p7	: Transport sap number
p8	: remote Transport selector
p9	: Transport channel number
p10	: Hexadecimal bitstring:

0ER0 ssss 0000 0TVV FFFF FFFF FFFF FFFF

E	: 1 = connection established
R	: 1 = incoming connection
T	: 1 = Texp in use
VV	: Session version number: 01=1, 10=2, or 11=either, based on negotiation
ssss	: state at time of message generation
FFFF	: functional units 2 bytes

P11 : Hexadecimal bitstring:

0000 0000 0000 000r uuuu uuuu pppp pppp

r	: origin of disconnect is remote
uuuu	: reason delivered to user
pppp	: reason used in protocol

ACTION Examine the status and cause parameters to determine the reason and corrective actions.

MESSAGE	`[6001] P_CTX file line #`
	`[....] ENT=p1 CTX=p1 CHN=p3 SAP=p4 SUF=p5 SUF=p6 SAP=p7 ADD=p8 ADD=p9 ADD=p10 ADD=p11 ADD=p12 ADD=p13 CHN=p14 DCS=p15 ACSE=p16 Sta=p17 Sta=p18`

CAUSE	This is the output from the otsop show or find command. It shows the current context status.

PARAMETERS

p1	:entity number
p2	:Presentation context number
p3	:Presentation channel number
p4	:Presentation sap number
p5	:local Presentation selector
p6	:remote Presentation selector
p7	:Session sap number
p8	:remote Session selector
p9	:local Session selector
p10	:remote Transport selector
p11	:local Transport selector
p12	:remote network address
p13	:local network address
p14	:Session channel number
p15	:cardinal of the DCS
p16	:Presentation context for ACSE
p17	:status

0ERO 00VV ssss ssss ssss ssss Ff00 Yyxx

E	:1 = connection was established
R	:1 = incoming connection
ssss	:state at time of message generation
VV	:if ACSE, session version (1,2, 1+2)
F	:1 = context management presentation functional unit
	:1 = context restoration presentation functional unit
xx	:1 = service provided at local sap:
	00 = Presentation
	01 = ACSE + Presentation
	11 = ROSE + ACSE + Presentation
Y	:1 = Raw data mode (not for ROSE)
y	:1 = x410-1984 mode

p18 :extended status

CCCC CCCC CCCC CCCC FFFF FFFF FFFF FFFF

CCCC	:session requirements
FFFF	:revised session requirements

ACTION None.

MESSAGE	**[6003] Protocol Error file line # [....] ENT=p1 CTX=p1 CHN=p3 SAP=p4 SUF=p5 SUF=p6 SAP=p7 ADD=p8 ADD=p9 ADD=p10 ADD=p11 ADD=p12 ADD=p13 CHN=p14 DCS=p15 ACSE=p16 Sta=p17 Sta=p18**
CAUSE	A protocol error has been detected in the Presentation layer or below.

PARAMETERS

p1	:entity number
p2	:Presentation context number
p3	:Presentation channel number
p4	:Presentation sap number
p5	:local Presentation selector
p6	:remote Presentation selector
p7	:Session sap number
p8	:remote Session selector
p9	:local Session selector
p10	:remote Transport selector
p11	:local Transport selector
p12	:remote network address
p13	:local network address
p14	:Session channel number
p15	:cardinal of the DCS
p16	:Presentation context for ACSE
p17	:status

0ERO 00VV ssss ssss ssss ssss Ff00 Yyxx

E	:1 = connection was established
R	:1 = incoming connection
ssss	:state at time of message generation
VV	:if ACSE, session version (1,2, 1+2)
F	:1 = context management presentation functional unit
f	:1 = context restoration presentation functional unit
xx	:1 = service provided at local sap:
	00 = Presentation
	01 = ACSE + Presentation
	11 = ROSE + ACSE + Presentation
Y	:1 = Raw data mode (not for ROSE)
y	:1 = x410-1984 mode

p18 :extended status

CCCC CCCC CCCC CCCC FFFF FFFF FFFF FFFF

CCCC	:session requirements
FFFF	:revised session requirements

ACTION Check the log file and the offending PDU to determine the cause and corrective action.

MESSAGE	`[6004] User Error file line #`
	`[....] ENT=p1 CTX=p1 CHN=p3 SAP=p4 SUF=p5 SUF=p6`
	`SAP=p7 ADD=p8 ADD=p9 ADD=p10 ADD=p11 ADD=p12 ADD=p13`
	`CHN=p14 DCS=p15 ACSE=p16 Sta=p17 Sta=p18`

CAUSE	A user has attempted an improper action.

PARAMETERS

p1	:entity number
p2	:Presentation context number
p3	:Presentation channel number
p4	:Presentation sap number
p5	:local Presentation selector
p6	:remote Presentation selector
p7	:Session sap number
p8	:remote Session selector
p9	:local Session selector
p10	:remote Transport selector
p11	:local Transport selector
p12	:remote network address
p13	:local network address
p14	:Session channel number
p15	:cardinal of the DCS
p16	:Presentation context for ACSE
p17	:status

0ERO 00VV ssss ssss ssss ssss Ff00 Yyxx

E	:1 = connection was established
R	:1 = incoming connection
ssss	:state at time of message generation
VV	:if ACSE, session version (1,2, 1+2)
F	:1 = context management presentation functional unit
f	:1 = context restoration presentation functional unit
xx	:1 = service provided at local sap:
	00 = Presentation
	01 = ACSE + Presentation
	11 = ROSE + ACSE + Presentation
Y	:1 = Raw data mode (not for ROSE)
y	:1 = x410-1984 mode

p18 :extended status

CCCC CCCC CCCC CCCC FFFF FFFF FFFF FFFF

CCCC	:session requirements
FFFF	:revised session requirements

ACTION Examine the connection state and error cause to determine the e corrective actions.

MESSAGE	**`[6007] P_ESTAB file line #`**
	`[....] ENT=p1 CTX=p1 CHN=p3 SAP=p4 SUF=p5 SUF=p6 SAP=p7 ADD=p8 ADD=p9 ADD=p10 ADD=p11 ADD=p12 ADD=p13 CHN=p14 DCS=p15 ACSE=p16 Sta=p17 Sta=p18`

CAUSE	Connection establishment is being attempted between two endpoints.

PARAMETERS		
	p1	:entity number
	p2	:Presentation context number
	p3	:Presentation channel number
	p4	:Presentation sap number
	p5	:local Presentation selector
	p6	:remote Presentation selector
	p7	:Session sap number
	p8	:remote Session selector
	p9	:local Session selector
	p10	:remote Transport selector
	p11	:local Transport selector
	p12	:remote network address
	p13	:local network address
	p14	:Session channel number
	p15	:cardinal of the DCS
	p16	:Presentation context for ACSE
	p17	:status

0ERO 00VV ssss ssss ssss ssss Ff00 Yyxx

E	:1 = connection was established
R	:1 = incoming connection
ssss	:state at time of message generation
VV	:if ACSE, session version (1,2, 1+2)
F	:1 = context management presentation functional unit
f	:1 = context restoration presentation functional unit
xx	:1 = service provided at local sap:
	00 = Presentation
	01 = ACSE + Presentation
	11 = ROSE + ACSE + Presentation
Y	:1 = Raw data mode (not for ROSE)
y	:1 = x410-1984 mode

p18 :extended status

CCCC CCCC CCCC CCCC FFFF FFFF FFFF FFFF

CCCC	:session requirements
FFFF	:revised session requirements

ACTION None.

MESSAGE	`[6008] P_REJECT file line #`
	`[....] ENT=p1 CTX=p1 CHN=p3 SAP=p4 SUF=p5 SUF=p6 SAP=p7 ADD=p8 ADD=p9 ADD=p10 ADD=p11 ADD=p12 ADD=p13 CHN=p14 DCS=p15 ACSE=p16 Sta=p17 Sta=p18`

CAUSE	A Presentation connect request has been rejected.

PARAMETERS

p1	:entity number
p2	:Presentation context number
p3	:Presentation channel number
p4	:Presentation sap number
p5	:local Presentation selector
p6	:remote Presentation selector
p7	:Session sap number
p8	:remote Session selector
p9	:local Session selector
p10	:remote Transport selector
p11	:local Transport selector
p12	:remote network address
p13	:local network address
p14	:Session channel number
p15	:cardinal of the DCS
p16	:Presentation context for ACSE
p17	:status

0ERO 00VV ssss ssss ssss ssss Ff00 Yyxx

E	:1 = connection was established
R	:1 = incoming connection
ssss	:state at time of message generation
VV	:if ACSE, session version (1,2, 1+2)
F	:1 = context management presentation functional unit
f	:1 = context restoration presentation functional unit
xx	:1 = service provided at local sap:
	00 = Presentation
	01 = ACSE + Presentation
	11 = ROSE + ACSE + Presentation
Y	:1 = Raw data mode (not for ROSE)
y	:1 = x410-1984 mode

p18 :extended status

CCCC CCCC CCCC CCCC FFFF FFFF FFFF FFFF

CCCC	:session requirements
FFFF	:revised session requirements

ACTION Examine the cause parameter to determine the reason and the corrective actions.

MESSAGE	`[6009] P_DCNX_OK file line #`
	`[....] ENT=p1 CTX=p1 CHN=p3 SAP=p4 SUF=p5 SUF=p6 SAP=p7 ADD=p8 ADD=p9 ADD=p10 ADD=p11 ADD=p12 ADD=p13 CHN=p14 DCS=p15 ACSE=p16 Sta=p17 Sta=p18`	

CAUSE	A normal Presentation connection disconnect has occurred.	
PARAMETERS	p1	:entity number
	p2	:Presentation context number
	p3	:Presentation channel number
	p4	:Presentation sap number
	p5	:local Presentation selector
	p6	:remote Presentation selector
	p7	:Session sap number
	p8	:remote Session selector
	p9	:local Session selector
	p10	:remote Transport selector
	p11	:local Transport selector
	p12	:remote network address
	p13	:local network address
	p14	:Session channel number
	p15	:cardinal of the DCS
	p16	:Presentation context for ACSE
	p17	:status

0ERO 00VV ssss ssss ssss ssss Ff00 Yyxx

E	:1 = connection was established
R	:1 = incoming connection
ssss	:state at time of message generation
VV	:if ACSE, session version (1,2, 1+2)
F	:1 = context management presentation functional unit
f	:1 = context restoration presentation functional unit
xx	:1 = service provided at local sap:
	00 = Presentation
	01 = ACSE + Presentation
	11 = ROSE + ACSE + Presentation
Y	:1 = Raw data mode (not for ROSE)
y	:1 = x410-1984 mode

p18 :extended status

CCCC CCCC CCCC CCCC FFFF FFFF FFFF FFFF

CCCC	:session requirements
FFFF	:revised session requirements

ACTION None.

MESSAGE	`[6010] P_DCNX_KO file line #`
	`[....] ENT=p1 CTX=p1 CHN=p3 SAP=p4 SUF=p5 SUF=p6 SAP=p7 ADD=p8 ADD=p9 ADD=p10 ADD=p11 ADD=p12 ADD=p13 CHN=p14 DCS=p15 ACSE=p16 Sta=p17 Sta=p18`

CAUSE	An abnormal Presentation connection disconnect has occurred.

PARAMETERS

p1	:entity number
p2	:Presentation context number
p3	:Presentation channel number
p4	:Presentation sap number
p5	:local Presentation selector
p6	:remote Presentation selector
p7	:Session sap number
p8	:remote Session selector
p9	:local Session selector
p10	:remote Transport selector
p11	:local Transport selector
p12	:remote network address
p13	:local network address
p14	:Session channel number
p15	:cardinal of the DCS
p16	:Presentation context for ACSE
p17	:status

0ERO 00VV ssss ssss ssss ssss Ff00 Yyxx

E	:1 = connection was established
R	:1 = incoming connection
ssss	:state at time of message generation
VV	:if ACSE, session version (1,2, 1+2)
F	:1 = context management presentation functional unit
f	:1 = context restoration presentation functional unit
xx	:1 = service provided at local sap: 00 = Presentation 01 = ACSE + Presentation 11 = ROSE + ACSE + Presentation
Y	:1 = Raw data mode (not for ROSE)
y	:1 = x410-1984 mode

p18 :extended status

CCCC CCCC CCCC CCCC FFFF FFFF FFFF FFFF

CCCC	:session requirements
FFFF	:revised session requirements

ACTION Examine the status and cause parameters to determine the reason and corrective actions.

MESSAGE

```
[8022] file line #
[....] U_MAP PID=p1 PATHID=p2 ENT=p3 CHN=p4
```

CAUSE Relates the user process's application connection id (PATHID) to the protocol stack's channel number (CHN). PATHID is unique to the user process, and CHN is unique to the OTS/9000 entity. The PATHID may be used to find corresponding log messages in the log file(s) for higher layers.

PARAMETERS

p1:	application process number
p2:	process connection id
p3:	protocol stack entity number
p4:	entity channel number

ACTION None.

MESSAGE

```
[8029]file line #
[....] Admin Register from: MBX <admin_mailbox>
```

CAUSE Administrative register request. An application has registered on the mailbox.

PARAMETERS admin_mailbox: name of application FIFO

ACTION None.

MESSAGE

```
[8029]file line #
[....] Admin Disregister from: MBX <admin_mailbox>
```

CAUSE Administrative register cancel. Application shutdown caused the process to deregister from the mailbox.

PARAMETERS admin_mailbox: name of application FIFO

ACTION None.

MESSAGE	`[8029]file line #`
	`[....] Command: <command_echo> MBX <admin_mailbox>`

CAUSE	Operator or administrative command received by the protocol stack.
PARAMETERS	command_echo: operator command admin_mailbox: name of application FIFO
ACTION	None.

MESSAGE	`[8029]file line #`
	`[....] ERROR detected  > sys_errlist[<errno>] for: <file_name>`

CAUSE	Open file error.
PARAMETERS	file_name: name of file on which error occurred
ACTION	Examine /usr/include/sys/errno.h to determine what system level error occurred.

MESSAGE	`[8029]file line #`
	`[....] OTSAM Bad interaction From <from_mbx> TO <to_mbx> I (<from_id>,<to_id>)`

CAUSE	Invalid interaction received. The product has received a corrupt or improperly formatted interaction. This may cause further product errors if recovery is not possible.
PARAMETERS	Internal reference numbers: from_mbx to_mbx from_id to_id
ACTION	Save log files and contact HP support if the problem persists.

MESSAGE	`[8029]file line #`
	`[....] OTSAM error MBX ir: <errno>`

CAUSE	Cannot read the common part of interaction from MBXOSI.
PARAMETERS	errno: of read
ACTION	None.

MESSAGE	`[8029]file line #`
	`[....] OTSAM error MBX ir lp: <lp>`

CAUSE	Cannot read interaction parameter because of incorrect length.
PARAMETERS	lp: length of interaction parameter
ACTION	None.

MESSAGE	`[8029]file line #`
	`[....] OTSAM error MBX irv: <errno>`

CAUSE	Cannot read interaction parameter.
PARAMETERS	errno: of readv
ACTION	None.

MESSAGE	`[8029]file line #`
	`[....] OTSAM error MBX read: <errno>`

CAUSE	Read of an interaction from a mailbox failed.
PARAMETERS	errno: of read
ACTION	None.

MESSAGE	`[8029]file line #`
	`[....] OTSAM error read on LAN : <errno>`

CAUSE	Error when reading from LAN.
PARAMETERS	errno: of read
ACTION	None.

MESSAGE

```
[8029] file line #
[....] OTSAM error sbm_drop <sbm_error> extbuf
<extbuf>
```

CAUSE

The shared memory buffer was not or could not be correctly released. While this may not be a serious problem for a single buffer or a connection, repeated errors will reduce the free shared memory space and applications may fail.

PARAMETERS

sbm_error returned by sbm_drop:

-11	: Association identifier is not valid
-13	: Data structure corruption
-17	: Address specified is not valid

extbuf: buffer to be released

ACTION

If this occurs, stop all applications and restart the stack. This will cause all shared memory to be returned and reallocated.

MESSAGE

```
[8029]file line #
[....] OTSAM error sbm_alc iw : <sbm_error>
```

CAUSE

Cannot allocate the required size in shared memory. This indicates that either there is insufficient shared buffer memory or an application is incorrectly addressing shared buffer memory.

PARAMETERS

sbm_error:

-11	: Association identifier is not valid.
-13	: Data structures are corrupted.
-15	: Size is not positive, or insufficient free memory is associated with the association identifier.

ACTION

Use the `-m` option on the *otsstart* command to increase the total allocated shared memory.

MESSAGE	`[8029]file line #`
	`[....] OTSAM error MBX iw : destination_mailbox, <errno>`

CAUSE	An attempt to write to a mailbox failed.
PARAMETERS	errno: of writev
ACTION	None.

MESSAGE	`[8029]file line #`
	`[....] OTSAM error sbm_alc x25osint : <sbm_error>`

CAUSE	Cannot allocate the required size. This indicates that either there is insufficient shared buffer memory or an application is incorrectly addressing shared buffer memory.
PARAMETERS	sbm_error: -11 : Association identifier is not valid. -13 : Data structures are corrupted. -15 : Size is not positive, or insufficient free memory associated with the association identifier.
ACTION	Use the `-m` option on the *otsstart* command to increase the total allocated shared memory.

MESSAGE	`[8029]file line #`
	`[....] OTSAM error LAN log_dest_addr <n> on <address> <errno>`

CAUSE	NETCTRL ioctl (request type: LOG_DEST_ADDR) failed.
PARAMETERS	n: return code from ioctl address: network destination address errno:
ACTION	None.

MESSAGE	`[8029]file line #`
	`[....] OTSAM error LAN iw on sn: <sap>, <errno>`

CAUSE	An attempt to write to the LAN failed.
PARAMETERS	sap: internal SAP number errno: of writev
ACTION	Execute diagnostics to determine problem area. If the problem exists in OTS/9000, gather log files and note the operating environment and contact HP support.

MESSAGE	`[8029]file line #`
	`[....] OTSAM error write uw <destination_mailbox> : <errno>`

CAUSE	An attempt to write to a mailbox failed.
PARAMETERS	errno: of writev destination_mailbox: internal reference number
ACTION	None.

MESSAGE	`[8029]file line #`
	`[....] OTSAM error open r mbx: <file_name>, <errno>`

CAUSE	An attempt to open a mailbox with read access failed.
PARAMETERS	file_name: name of FIFO errno: of open
ACTION	Verify that the mailbox permissions are correctly set when the application or OTS/9000 is in operation. Mail boxes are usually created dynamically on a per-process basis.

MESSAGE	**`[8029]file line #`**
	`[....] OTSAM error open w mbx: <file_name>, <errno>`

CAUSE	An attempt to open a mailbox with write access failed.
PARAMETERS	file_name: name of FIFO errno: of open
ACTION	Verify that the mailbox permissions are correctly set when the application or OTS/9000 is in operation. Mail boxes are usually created dynamically on a per-process basis.

MESSAGE	**`[8029]file line #`**
	`[....] OTSAM error open close mbx: <mailbox_pointer>`
	`, <errno>`

CAUSE	Close of mailbox failed. Mail boxes are usually created dynamically and are usually closed and removed after each usage.
PARAMETERS	mailbox_pointer: name of mailbox errno: of close
ACTION	Remove the mailbox with the rm(1) command.

MESSAGE	**`[8029]file line #`**
	`[....] OTSAM error x25 init : <status>`

CAUSE	Open of the X.25 provider access method failed.
PARAMETERS	status: return code of x25_init
ACTION	Verify X.25 is operational before starting the OSI product.

MESSAGE	**`[8029]file line #`**
	`[....] OTSAM error open LAN log_ssap <<n> on <arg.value.i>  <errno>`

CAUSE	NETCTRL ioctl (request type: LOG_SSAP) failed while opening LAN access method.
PARAMETERS	n: return of ioctl arg.value.i: passed to ioctl errno:
ACTION	Verify the LAN is operational. This is an informational message which might be corrected by restarting the OSI products.

MESSAGE	**`[8029]file line #`**
	`[....] OTSAM error open LAN log_read_cache <n> <errno>`

CAUSE	NETCTRL ioctl (request type: LOG_READ_CACHE) failed while opening LAN access method.
PARAMETERS	n: return of ioctl errno:
ACTION	This is an informational message which might be corrected by restarting the OSI products.

MESSAGE	**`[8029]file line #`**
	`[....] OTSAM error open LAN add_multicast <n> for all-is  <errno>`

CAUSE	NETCTRL ioctl (request type: ADD_MULTICAST) failed while opening LAN access method.
PARAMETERS	n: return of ioctl errno:
ACTION	This is an informational message which might be corrected by restarting the OSI products.

MESSAGE	`[8029]file line #` `[....] OTSAM error open LAN add_multicast n for all-es <errno>`
CAUSE	NETCTRL ioctl (request type: ADD_MULTICAST) failed while opening LAN access method.
PARAMETERS	n: return of ioctl errno:
ACTION	This is an informational message which might be corrected by restarting the OSI products.

MESSAGE	`[8029]file line #` `[....] OTSAM error open LAN <sap> service not CLSNS`
CAUSE	An internal error indicating that the service has not been configured for connectionless usage.
PARAMETERS	SAP
ACTION	Gather log files, note operating environment and contact HP support.

MESSAGE	`[8029]file line #` `[....] OTSAM Shutdown in progress ...`
CAUSE	OTSAM shutdown in progress. Do not attempt to stop the shutdown - it cannot be stopped until clean up is complete.
ACTION	None.

MESSAGE	`[8029]file line #` `[....] OTSAM Shutdown complete ...`
CAUSE	OTSAM shutdown complete. The operator may now make any updates or modifications at this time. User applications utilizing the stack cannot operate.
ACTION	None.

MESSAGE	`[8029]file line #`
	`[....] OTSAM start otstic failed : <errno>`

CAUSE	Start of timer process failed. otstic is necessary for correct product operation.
PARAMETERS	errno: returned by fork / execl
ACTION	Correct the problem and restart the product.

MESSAGE	`[8029]file line #`
	`[....] UNKNOWN ERROR detected for: <file_name>`

CAUSE	Open file error. The product was unable to determine what error occurred.
PARAMETERS	file_name
ACTION	Examine the log file for possible problems.

MESSAGE	`[8030] file line #`
	`[....] Command: <command_echo> MBX <admin_mailbox>`

CAUSE	Invalid operator or administrative command was received.
PARAMETERS	command_echo admin_mailbox
ACTION	Examine the syntax error and take corrective action.

MESSAGE

```
[9500] file line #
[....] YYMMDD   hh:mm:ss.ssssss
```

YY (2 digits) is the year. For example, 89.

MM (2 digits) is the month. For example, 12.

DD (2 digits) is the day. For example, 05.

hh (2 digits) is the hour (24-hour format). For example, 17.

mm (2 digits) is the minute. For example, 30.

ss (8 digits) is the second to microsecond resolution. For example, 15.000113.

MESSAGE

```
[9901] file line #
[....] name PARM[num] (min..cur..max)
```

CAUSE

Output from otsop show entity command. Displays parameter information.

PARAMETERS

name	: name of the parameter being examined
num	: parameter number
min	: minimum acceptable value
cur	: current value
max	: maximum acceptable value

ACTION

None.

MESSAGE

```
[9902] file line #
[....] name PARM[num] (min..cur..max)
```

CAUSE

Output from otsop set parameter command. Displays updated parameter information.

PARAMETERS

name	: name of the parameter being set
num	: parameter number
min	: minimum acceptable value
cur	: current value
max	: maximum acceptable value

ACTION

None.

MESSAGE	`[9903] file line #`
	`[....] Buffer(data) tot=p1 free=p2 size=p3 qt=p4`

CAUSE	Output from otsop show buffers command. Displays information about the data segment pool.
PARAMETERS	p1 : total number of segments ever allocated p2 : number of free data segments p3 : size of data segment p4 : allocation quantum
ACTION	None.

MESSAGE	`[9904] file line #`
	`[....] Buffer (ctrl) tot=01 free=p2 qt=p3`

CAUSE	Output from otsop show buffers command. Displays information about the data segment pool.
PARAMETERS	p1 : total number of segments ever allocated p2 : number of free data segments p3 : size of data segment
ACTION	None.

MESSAGE	`[9905] file line #`
	`[....] ENT=p1, CTX=(F=p2, B=p3), CHN=(F=p4, B=p5), STA=p6`

CAUSE	Output from otsop show entity command. Displays entity resources information.
PARAMETERS	p1 : entity number p2 : number of free contexts p3 : number of contexts used p4 : number of free channels p5 : number of channels used p6 : entity state (c = closed, A = active)
ACTION	None.

MESSAGE	`[9906] file line #`
	`[....] CHN(p1) STA=A(p2) HX=(p3,p4) LX=(p5,p6)`

CAUSE	Output from otsop show channel command. Displays channel information.
PARAMETERS	p1 : channel number p2 : channel state p3 : entity using the channel p4 : context attached as a user p5 : entity providing service at this channel p6 : context attached as a provider
ACTION	None.

MESSAGE	`[9907] file line #`
	`[....] SAP=p1 MAP=(p2,p3,p4) STA=p5 HE=p6 LE=p7`
	`SR=p8 AM=(p9,p10) SUF=p11 PRI=p12 CHN=p13`
	`LSAP=(p14,p15,p16) Feat=p17 TRACE=p18`

PARAMETERS	p1 : internal SAP number (decimal) p2 : SAP at the next lower layer (decimal) (-1 = none) p3 : First SAP at the next higher layer (decimal) (-1 = none) p4 : Next SAP that has the same entity below (decimal) (-1 = none) or next SAP mapped onto the same next lower SAP p5 : SAP state (decimal) 0 - on 1 - off 2 - this is a dummy SAP p6 : Entity above this SAP (decimal) 80 - user library (API) 50 - Session, 40 - Transport over CONS 41 - Transport over CLNS method)

p7	: Entity below this SAP (decimal) 50 - Session 40 - Transport over CONS 41 - Transport over CLNS 00 - low access method
p8	: Service provided by SAP (decimal) 0 - dummy 1 - CONS 2 - CLNS 3 - CLSNS 7 - Transport 8 - Session
p9	: High access method (decimal) 0 - Reserved (internal access) 1 - user library (API)
p10	: Low access method (decimal) 0 - Reserved (internal access) 1 - X.25 access method 2 - demonstration access method 3 - LAN access method
p11	: SAP selector (hexadecimal; first byte is the length)
p12	: Private data for this sap (hexadecimal)
p13	: Number of channels attached to this SAP
p14	: Next Lateral SAP (-1 = none)
p15	: Lateral SAP flag 0 - Regular SAP 1 - Lateral SAP
p16	: Reserved
p17	: Reserved
p18	: Trace status (T = enabled, t = disabled)

MESSAGE	`[9908] file line #`
	`[....] ENT=p1, CHN=p2, USR=p3, TIME=p4, RCV=p5, SND=p6`

CAUSE	Gives channel accounting information which closed and had accounting enabled.
PARAMETERS	p1 : entity number p2 : channel number p3 : user id p4 : time elapsed p5 : number of bytes received p6 : number of bytes sent
ACTION	None.

MESSAGE	`[9909] file line #`
	`[....] EVN=p1 CHN=p2 CTX=p3 ENT#=p4`

CAUSE	Display interaction information delivered to a channel which has debug or trace set.
PARAMETERS	p1 : interaction code: 1xxx = inbound 2xxx = outbound p2 : channel number p3 : context to which the interaction is directed p4 : entity to which the interaction is directed.
ACTION	None.

MESSAGE	`[9910] file line #`
	`[....] SAP=p1 Sap set to:p2`

CAUSE	Output from otsop vary sap command. Displays new status.
PARAMETERS	p1 : SAP number p2 : S = SAP active, s = SAP inactive
ACTION	None.

MESSAGE	`[9911] file line #`
	`[....] CHN=p1 Trace set to:p2`

CAUSE	Output from otsop vary entity trace command. Displays new channel status.
PARAMETERS	p1 : Channel number p2 : T = trace active, t = trace inactive
ACTION	None.

MESSAGE	`[9912] file line #`
	`[....] CTX(p1) STA=A(p2)`

CAUSE	Output from ostop show entity context command. Displays context status.
PARAMETERS	p1 : context number p2 : state number
ACTION	If the entity supports the "show" function, the message will be followed by a more comprehensive display done.

MESSAGE	`[9913] Xfer file line #`
	`[....] EVN=p1 CHN=p2 Sap=p3`

CAUSE	Displays interaction information when it is placed on a channel that has debug or trace enabled.
PARAMETERS	p1 : interaction code: 1xxx = inbound 2xxx = outbound p2 : channel number p3 : internal SAP number
ACTION	None.

MESSAGE	`[9914] ENT=p1, Invalid Parameter file line #`
	`[....] p2`

CAUSE	Output from otsop when the operator attempts to set an entity parameter that is out of range or the parameter number does not exist. The command is not executed.
PARAMETERS	p1 : entity number p2 : parameter number
ACTION	None.

MESSAGE	`[9915] Inv. Ent # file line #`
	`[....] p1`

CAUSE	Output from otsop when the operator attempts to issue a command to a non-existing entity.
PARAMETERS	p1 : entity number
ACTION	None.

MESSAGE	`[9916] Debug set to: file line #`
	`[....] p1`

CAUSE	Output from otsop debug command. Displays result.
PARAMETERS	p1 : D = on, d = off
ACTION	None.

MESSAGE	`[9917] Inv. sap # file line #`
	`[....] p1`

CAUSE	Output from otsop when the operator attempts to issue a command to a non-existing sap.
PARAMETERS	p1 : SAP number
ACTION	None.

MESSAGE	`[9920] Inv. vary option file line #`
	`[....] p1`

CAUSE	Output from otsop vary command containing an invalid component name.
PARAMETERS	p1 : component name
ACTION	None.

MESSAGE	`[9921] Inv. show option file line #`
	`[....] p1`

CAUSE	Output from otsop show command containing an invalid component name.
PARAMETERS	p1 : component name
ACTION	None.

MESSAGE	`[9922] Inv. general command file line #`
	`[....] p1`

CAUSE	Output from otsop when an invalid command is issued.
PARAMETERS	p1 : command name
ACTION	None.

MESSAGE	`[9923] file line #`
	`[....] CHN(p1) STA=Inactive`

CAUSE	Output from otsop show channel command. Displays channel status.
PARAMETERS	p1 : channel number
ACTION	None.

MESSAGE	`[9924] Invalid channel # file line #`
	`[....] p1`	

CAUSE	Output from otsop when a non-existent channel is addressed.	
PARAMETERS	p1	: channel number
ACTION	None.	

MESSAGE	`[9925] file line #`
	`[....] CTX(p1) STA=Inactive`	

CAUSE	Output from otsop show context command. Displays context status.	
PARAMETERS	p1	: context number
ACTION	None.	

MESSAGE	`[9926] Inv. context # file line #`
	`[....] p1`	

CAUSE	Output from otsop when a non-existent context is addressed.	
PARAMETERS	p1	: context number
ACTION	None.	

MESSAGE	`[9927] Entity Title = file line #`
	`[....] p1`	

CAUSE	Entity title.	
PARAMETERS	p1	: title
ACTION	None.	

MESSAGE	**[9928] Cancel executed file** **[....] (p1)**
CAUSE	A cancel connection command has been issued. Will only be displayed if the entity has the "cancel connection" facility.
PARAMETERS	p1 : channel number
ACTION	None.

MESSAGE	**[9929] Dynamic - SAP file** **[....] Mount or Mark p1**
CAUSE	Output from otsop when a non-existent channel is addressed.
PARAMETERS	p1 : Mount or mark error
ACTION	None.

MESSAGE	**[9930] Restart of entity # Ent file** **[....] p1**
CAUSE	Output from otsop vary entity command.
PARAMETERS	p1 : entity number
ACTION	None.

MESSAGE	**[9931] Shutdown in progress file** **[....] p1**
CAUSE	Output from otsop vary entity off/force command.
PARAMETERS	p1 : entity number
ACTION	None.

MESSAGE	**`[9932] Fatal error file`**
	`[....] <err_number>, <message>`

CAUSE	OTSAM detected a catastrophic error from which it could not recover. Core dump may be produced.
PARAMETERS	see table below

err_number	message
0	Length error
1	No control block available
2	No memory for buffer control block allocation
3	Unable to find layer entity control block
4	No channel
5	No memory for buffer DE allocation
6	No memory for initial control block allocation
7	TU error
8	Release/Locate Error
9	Channel Error
10	Magic number mismatch
11	Memory mngmt error
12	Layer error code

ACTION Save the core file and contact HP support for assistance. Also, save the OTS/9000 and application log file(s).

MESSAGE	**`[9933] Inv. specific command file`**
	`[....] p1`

CAUSE	Output from otsop indicating an invalid command issued.
PARAMETERS	p1 : command name
ACTION	None.

MESSAGE	**`[9934] Shutdown complete file`**
	`[....] p1`

CAUSE	Entity shutdown has completed as a result of an otsop vary entity off/force command.
PARAMETERS	entity number
ACTION	None.

MESSAGE	`[9935] Unauthorized command file`
	`[....]`

CAUSE	A command was issued in an unauthorized mode.
ACTION	None.

MESSAGE	`[9936] Intcb # modified file`
	`[....]`

CAUSE	Indicates that the otsop define interaction command completed.
ACTION	None.

MESSAGE	`[9937] Inv. sap or incompatible suffix file`
	`[....]`

CAUSE	As a result of a define SAP command, the command cannot be executed.
ACTION	Redefine the SAP.

MESSAGE	`[9938] file`
	`[....] Buffer(memd) tot=p1 free=p2 (frgd) tot=p3 free=p4`

CAUSE	Output from otsop show buffer command.
PARAMETERS	p1 : total number of memory descriptors p2 : number of free memory descriptors p3 : total number of fragment descriptors p4 : number of free fragment descriptors
ACTION	None.

MESSAGE	`[9940] Prefix set file`
	`[....]`

CAUSE	Output from otsop set prefix command.
ACTION	None.

MESSAGE	`[9941] Cancel not supported file`
	`[....]`

CAUSE	Output from otsop cancel connection command. Indicates that the entity does not provide for the cancel connection facility.
ACTION	None.

MESSAGE	`[9942] Started file`
	`[....]`

CAUSE	Output from otsop start command indicating the stack has successfully started.
ACTION	None.

MESSAGE	`[9943] file`
	`[....] macro p1 is p2`

CAUSE	Output from otsop macro command. Displays command results.
PARAMETERS	p1 : the macro name p2 : macro definition
ACTION	None.

MESSAGE	`[9944] FLAGS= file`
	`[....] p1`

CAUSE	Output from otsop show entity all command.
PARAMETERS	p1 : one flag group (8 flags); "_" = off, "X" = on
ACTION	None.

MESSAGE	`[9945] Flag = file`
	`[....] p1`

CAUSE	Output from otsop vary entity flag command.
PARAMETERS	p1 : one flag group (8 flags)
ACTION	None.

MESSAGE	`[9946] Inv. flag file`
	`[....]`	

CAUSE	Output from otsop vary entity flag command. Displays the invalid flag specified.	
ACTION	None.	

MESSAGE	`[9947] file`
	`[....] ADD=p1`	

CAUSE	Output from otsop show address command.	
PARAMETERS	p1	: address in hexadecimal
ACTION	None.	

MESSAGE	`[9948] Bad Parameter Value file`
	`[....]`	

CAUSE	An attempt was made to set a parameter, but the value is unacceptable.	
ACTION	None.	

MESSAGE	`[9949] Debug buffer trace file`
	`[....] p1 p2`	

CAUSE	Output from otsop debug mode command. Display the new trace characteristics.	
PARAMETERS	p1	: "y" if subchain(s) are traced, "n" otherwise
	p2	: maximum number of octets traced
ACTION	None.	

MESSAGE	`[9950] Buffer(s) in use: file`
	`[....] p1`	

CAUSE	Output from otsop show used buffer command by an entity.	
PARAMETERS	p1	: number of buffers.
ACTION	None.	

MESSAGE	`[9973] N-CONNECT-Cnf file`
	`[....] Bufferl is User data Responding N-address=p1 option=p2`	

CAUSE	Trace message that is written when the CONS entity is being traced and it receives a Network Connect Confirmation.	
PARAMETERS	p1	: called NSAP (hexadecimal; first byte is the length)
	p2	: options (in hexadecimal)
ACTION	None.	

MESSAGE	`[9973] N-CONNECT-Ind file`
	`[....] Bufferl is User data Local Sap=p1 Remote N-address=p2 user data=p3 option=p4 Local N-address=p5`	

CAUSE	Trace message that is written when the CONS entity is being traced and it receives a Network Connect Indication.	
PARAMETERS	p1	: local internal SAP number (hexadecimal)
	p2	: destination NSAP (hexadecimal; first byte is the length)
	p3	: user data (hexadecimal; first byte is the length)
	p4	: options (in hexadecimal)
	p5	: local NSAP, if sent in the PDU (hexadecimal; first byte is the length)
ACTION	None.	

MESSAGE	`[9973] N-CONNECT-Req file` `[....] Bufferl is User data Local Sap=p1 Remote N-address=p2 user data=p3 option=p4 Local N-address=p5`
CAUSE	Trace message that is written when the CONS entity is being traced and it sends a Network Connect Request.
PARAMETERS	p1 : local internal SAP number p2 : destination NSAP (hexadecimal; first byte is the length) p3 : user data (hexadecimal; first byte is the length) p4 : options (in hexadecimal) p5 : local NSAP, if sent in the PDU (hexadecimal; first byte is the length)
ACTION	None.

MESSAGE	`[9973] N-CONNECT-Rsp file` `[....] Bufferl is User data Responding N-address=p1 option=p2`
CAUSE	Trace message that is written when the CONS entity is being traced and it sends a Network Connect Response.
PARAMETERS	p1 : local NSAP, if sent in the PDU (hexadecimal; first byte is the length) p2 : options (in hexadecimal)
ACTION	None.

MESSAGE	`[9973] N-DATA-Ind file` `[....] Bufferl is User Data`
CAUSE	Trace message that is written when the CONS entity is being traced and it receives a Network Data Indication. The contents of the NPDU is shown in the subsequent message 9974.
ACTION	None.

MESSAGE	**[9973] N-DATA-Req file** **[....] Buffer1 is User Data**
CAUSE	Trace message that is written when the CONS entity is being traced and it sends a Network Data Request. The contents of the NPDU is shown in the subsequent message 9974.
ACTION	None.

MESSAGE	**[9973] N-DISCONNECT-Ind file** **[....] Buffer1 is User data Reason=p1 Responding address=p2**
CAUSE	Trace message that is written to when the CONS entity is being traced and it receives a Network Disconnect.
PARAMETERS	p1 : reason code (hexadecimal) p2 : called NSAP, if sent in the PDU (hexadecimal; first byte is the length)
ACTION	Take action based on the reason code. Refer to Troubleshooting X.25/9000.

MESSAGE	**[9973] N-DISCONNECT-Req file** **[....] Reason=p1 Responding address=p2**
CAUSE	Trace message that is written when the CONS entity is being traced and it sends a Network Disconnect.
PARAMETERS	p1 : reason code (hexadecimal) p2 : called NSAP, if sent in the PDU (hexadecimal; first byte is the length)
ACTION	Take action based on the reason code. Refer to Troubleshooting X.25/9000.

MESSAGE	**[9973] N-READY-Ind file** **[....]**
CAUSE	Internal flow-control between the CONS and Transport layers. The trace message is written when the CONS entity is being traced and it is ready to accept a TPDU from the Transport layer.
ACTION	None.

MESSAGE	**[9973] N-READY-Req file** **[....]**
CAUSE	Internal flow-control between the CONS and Transport layers. The trace message is written when the CONS entity is being traced and the Transport layer indicates that it is ready to receive a TPDU from the Network layer.
ACTION	None.

MESSAGE	**[9973] N-UNIT-DT-Indication file** **[....] Bufferl is User data Local Sap=pl** **Remote N-address=p2 option=p3** **Local N-address=p4**
CAUSE	Trace message that is written when the CLNS entity is being traced and it receives a Network Data Indication. The contents of the NPDU is shown in the subsequent message 9974. If the NPDU begins with hexadecimal 82, it is an ES-IS protocol message that was multicast from a system on the subnetwork.
PARAMETERS	p1 : internal Network SAP number (hexadecimal) p2 : remote MAC address (hexadecimal; first byte is the length) p3 : options (in hexadecimal) p4 : local NSAP, if sent in the PDU (hexadecimal; first byte is the length)
ACTION	None.

MESSAGE	**`[9973] N-UNIT-DT-Request file`**
	`[....] Bufferl is User data Local Sap=p1 Remote N-address=p2 option=p3 Local N-address=p4`

CAUSE	Trace message that is written when the CLNS entity is being traced and it sends a Network Data Request. The contents of the NDPU is shown in the subsequent message 9974.
PARAMETERS	p1 : internal Network SAP number (hexadecimal) p2 : remote MAC address (hexadecimal; first byte is the length) p3 : options (in hexadecimal) p4 : local NSAP, if sent in the PDU (hexadecimal; first byte is the length)
ACTION	None.

MESSAGE

```
[9973] S-CONNECT-Cnf file
[....] Buffer1 is Session Id Buffer2 is User Data
Size=p1
Init Serial number=p2 Init tk pos=p3 Service=p4
Please tk=p5
Functional Units=p6 Capability=p7
```

CAUSE

Trace message that is written when a Session Connect Confirmation is received on a Session SAP that is being traced. This may be followed by message 9974.

PARAMETERS

p1 : negotiated TSDU size (hexadecimal)

p2 : initial serial number for Session activity (hexadecimal)

p3 : initial token settings; hexadecimal, representing the bit string rrmmssdd, where:

rr = release token
mm = major/activity token
ss = synchronize-minor token
dd = data token

The values of each bit pair is:

00 = initiator's side
01 = responder's side
10 = responder's choice

p4 : internal Session service number (hexadecimal)

p5 : token item, indicating which tokens are requested by the responder, if the corresponding bit is 1; hexadecimal, representing the bit string rmsd0000, where:

r	= release token
m	= major/activity token
s	= synchronize-minor token
d	= data token

	p6:	negotiated functional units, if the corresponding bit is 1; hexadecimal, representing the bit string (bit 1 = least-significant bit):

16-15	= Session version 00 = version 1 10 = version 2
14	= extended concatenation
13-12	= Transport priority
11	= typed data
10	= exception reporting
9	= capability data
8	= negotiated release
7	= activity management
6	= resynchronize
5	= major synchronize
4	= minor synchronize
3	= expedited data
2	= full-duplex
1	= half-duplex

p7 : capability (hexadecimal)

ACTION None.

MESSAGE

```
[9973] S-CONNECT-Conf(Negative) file
[....] Bufferl is Session Id Buffer2 is User Data
Service=p1 Reason=p2 Capability=p3
```

CAUSE Trace message that is written when a Session Refuse is received through a Session SAP that is being traced. This may be followed by two 9974 messages.

PARAMETERS

p1	: internal Session service number (hexadecimal)
p2	: reason code (hexadecimal)
p3	: capability (hexadecimal)

ACTION Take action based on the reason code.

MESSAGE

```
[9973] S-CONNECT-Ind file
[....] Buffer1 is Session Id Buffer2 is User Data
Sap=p1 Size=p2 Init Serial number=p3 Init tk pos=p4
Service=p5 Functional Units=p6
SSAP Addr=p7 ..=p8 ..=p9 Capability=p10
```

CAUSE

Trace message that is written when a Session Connect Indication is received through a Session SAP that is being traced. This may be followed by two 9974 messages.

PARAMETERS

p1	: internal Session SAP number (hexadecimal)
p2	: proposed TSDU size (hexadecimal)
p3	: initial serial number for Session activity (hexadecimal)
p4	: initial token settings; hexadecimal, representing the bit string rrmmssdd, where: rr = release token; mm = major/activity token; ss = synchronize-minor token; dd = data token. The values of each bit pair is: 00 = initiator's side; 01 = responder's side; 10 = responder's choice
p5	: internal Session service number (hexadecimal)

	p6	: proposed functional units, if the corresponding bit is 1; hexadecimal, representing the bit string (bit 1 = least-significant bit):
		16-15 = Session version 00 = propose version 1 10 = propose version 2 11 = propose version 1 or 2
		14 = extended concatenation
		13-12 = Transport priority
		11 = typed data
		10 = exception reporting
		9 = capability data
		8 = negotiated release
		7 = activity management
		6 = resynchronize
		5 = major synchronize
		4 = minor synchronize
		3 = expedited data
		2 = full-duplex
		1 = half-duplex
	p7	: destination Session selector (hexadecimal; first byte is the length)
	p8	: destination Transport selector (hexadecimal; first byte is the length)
	p9	: destination NSAP (hexadecimal; first byte is the length)
	p10	: capability (hexadecimal)
ACTION	None.	

MESSAGE

```
[9973] S-CONNECT-Req file
[....] Buffer1 is Session Id Buffer2 is User Data
Sap=p1 Mode=p2 Init Serial number=p3 Init tk pos=p4
Service=p5 Functional Units=p6
SSAP Addr=p7 ..=p8 ..=p9 Capability=p10
```

CAUSE

Trace message that is written when a Session Connect Request is sent through a Session SAP that is being traced. This may be followed by two 9974 messages.

PARAMETERS

p1 : internal Session SAP number (hexadecimal)
p2 : mode (hexadecimal)
p3 : initial serial number for Session activity (hexadecimal)
p4 : initial token settings; hexadecimal, representing the bit string rrmmssdd, where:
  rr = release token
  mm = major/activity token
  ss = synchronize-minor token
  dd = data token
The values of each bit pair is:
  00 = initiator's side
  01 = responder's side
  10 = responder's choice
p5 : internal Session service number (hexadecimal)
p6 : proposed functional units, if the corresponding bit is 1; hexadecimal, representing the bit string (bit 1 = least-significant bit):
 16-15 = Session version
  00 = propose version 1
  01 = propose version 1
  10 = propose version 2
  11 = propose version 1 or 2

14	= extended concatenation
13-12	= Transport priority
11	= typed data
10	= exception reporting
9	= capability data
8	= negotiated release
7	= activity management
6	= resynchronize
5	= major synchronize
4	= minor synchronize
3	= expedited data
2	= full-duplex
1	= half-duplex

p7	: destination Session selector (hexadecimal; first byte is the length)
p8	: destination Transport selector (hexadecimal; first byte is the length)
p9	: destination NSAP (hexadecimal; first byte is the length)
p10	: capability (hexadecimal)

ACTION None.

MESSAGE	`[9973] S-CONNECT-Resp(Negative) file`
	`[....] Bufferl is Session Id Buffer2 is User Data`
	`Service=p1 Reason=p2 Capability=p3`

CAUSE	Trace message that is written when a Session Refuse is sent through a Session SAP that is being traced. This may be followed by two 9974 messages.
PARAMETERS	p1 : internal Session service number (hexadecimal) p2 : reason code (hexadecimal) p3 : capability (hexadecimal)
ACTION	Take action based on the reason code.

MESSAGE	`[9973] S-CONNECT-Rsp file`
	`[....] Bufferl is Session Id Buffer2 is User Data`
	`Mode=p1`
	`Init Serial number=p2 Init tk pos=p3 Service=p4`
	`Please tk=p5`
	`Functional Units=p6 Capability=p7`

CAUSE	Trace message that is written when a Session Connect Response is sent through a Session SAP that is being traced. This may be followed by two 9974 messages.
PARAMETERS	p1 : mode (hexadecimal) p2 : initial serial number for Session activity (hexadecimal) p3 : initial token settings; hexadecimal, representing the bit string rrmmssdd, where: rr = release token mm = major/activity token ss = synchronize-minor token dd = data token The values of each bit pair is: 00 = initiator's side 01 = responder's side 10 = responder's choice p4 : internal Session service number (hexadecimal)

p5 : token item, indicating which tokens are requested by the responder, if the corresponding bit is 1; hexadecimal, representing the bit string rmsd0000, where:

r = release token
m = major/activity token
s = synchronize-minor token
d = data token

p6 : negotiated functional units, if the corresponding bit is 1; hexadecimal, representing the bit string (bit 1 = least-significant bit):

bit 16-15 = Session version

00 = version 1
01 = version 1
10 = version 2
11 = version 1 or 2

14	= extended concatenation
13-12	= Transport priority
11	= typed data
10	= exception reporting
9	= capability data
8	= negotiated release
7	= activity management
6	= resynchronize
5	= major synchronize
4	= minor synchronize
3	= expedited data
2	= full-duplex
1	= half-duplex

p7 : capability (hexadecimal)

ACTION None.

MESSAGE	`[9973] S-DATA-Ind file`
	`[....] Buffer2 is User Data pril=pl Tk=p2`

CAUSE	Trace message that is written when a Session Data Indication is received through a Session SAP that is being traced. This is followed by message 9974.
PARAMETERS	p1 : priority (hexadecimal) p2 : token (hexadecimal)
ACTION	None.

MESSAGE	`[9973] S-DATA-Req file`
	`[....] Buffer2 is User Data pril=pl Tk=p2`

CAUSE	Trace message that is written when a Session Data Request is sent through a Session SAP that is being traced. This is followed by message 9974.
PARAMETERS	p1 : priority (hexadecimal) p2 : token (hexadecimal)
ACTION	None.

MESSAGE	`[9973] S-P-ABORT-Ind file`
	`[....] Reason=pl`

CAUSE	Trace message that is written when a Session Provider Abort is sent locally through a Session SAP that is being traced.
PARAMETERS	p1 : reason code (hexadecimal)
ACTION	Take action based on the reason code.

MESSAGE	`[9973] S-READY-Ind file`
	`[....]`

CAUSE	Internal flow-control between the Session layer and the user's application. The trace message is written when the Session SAP is being traced and it is ready to accept a PPDU from the user's application.
ACTION	None.

MESSAGE
```
[9973] S-READY-Req file
[....]
```
CAUSE Internal flow-control between the Session layer and user's application. The trace message is written when the Session SAP is being traced and the user's application indicates that it is ready to receive a PPDU from the Session layer.

ACTION None.

MESSAGE
```
[9973] S-RELEASE-Cnf file
[....] Buffer2 is User Data
```
CAUSE Trace message that is written when a Session Release Confirmation is received through a Session SAP that is being traced. This may be followed by message 9974.

ACTION None.

MESSAGE
```
[9973] S-RELEASE-Ind file
[....] Buffer2 is User Data
```
CAUSE Trace message that is written when a Session Release Indication is received through a Session SAP that is being traced. This may be followed by message 9974.

ACTION None.

MESSAGE
```
[9973] S-RELEASE-Req file
[....] Buffer2 is User Data
```
CAUSE Trace message that is written when a Session Release Request is sent through a Session SAP that is being traced. This may be followed by a message 9974.

ACTION None.

MESSAGE	`[9973] S-RELEASE-Rsp file`
	`[....] Buffer2 is User Data`

CAUSE	Trace message that is written when a Session Release Response is sent through a Session SAP that is being traced. This may be followed by a message 9974.
ACTION	None.

MESSAGE	`[9973] T-CONNECT-Cnf file`
	`[....] Buffer1 is User Data Class/Options=p1 Size=p2`

CAUSE	Trace message that is written when a Transport Connect Confirmation is received through a Transport SAP that is being traced. This may be followed by message 9974.

PARAMETERS	p1 : negotiated options (hexadecimal):
	a0 = TP4 for CLNS, or TP2 with alternate TP0 for CONS
	00 = TP0
	p2 : negotiated TPDU size (hexadecimal)

ACTION	None.

MESSAGE	**[9973] T-CONNECT-Ind file** **[....] Bufferl is User Data Local Sap=p1 Size=p2** **Class/Options=p3 Qos=p4** **Remote T-address=p5 ..=p6** **Local T-address=p7 ..=p8**
CAUSE	Trace message that is written when a Transport Connect Indication is receive through a Transport SAP that is being traced. This may be followed by message 9974.
PARAMETERS	p1 : internal Transport SAP number (hexadecimal) p2 : TPDU size (hexadecimal) p3 : options (hexadecimal): a0 = TP4 for CLNS, or TP2 with alternate TP0 for CONS 00 = TP0 p4 : quality of service (hexadecimal) p5 : destination Transport selector (hexadecimal; first byte is the length) p6 : destination NSAP (hexadecimal; first byte is the length) p7 : local Transport selector, if sent in the PDU (hexadecimal; first byte is the length) p8 : local NSAP, if sent in the PDU (hexadecimal; first byte is the length)
ACTION	None.

MESSAGE	**[9973] T-CONNECT-Rsp file** **[....] Bufferl is User Data Class/Options=p1 Size=p2**
CAUSE	Trace message that is written when a Transport Connect Response is sent through a Transport SAP that is being traced. This may be followed by message 9974.

PARAMETERS	p1 : negotiated options (hexadecimal):
	a0 = TP4 for CLNS, or TP2 for CONS
	00 = TP0
	p2 : negotiated TPDU size (hexadecimal)

ACTION	None.

MESSAGE	`[9973] T-CONNECT-Req file`
	`[....] Buffer1 is User Data Local Sap=p1 Size=p2`
	`Class/Options=p3 Qos=p4`
	`Remote T-address=p5 ..=p6`
	`Local T-address=p7 ..=p8`

CAUSE	Trace message that is written when a Transport Connect Request is sent through a Transport SAP that is being traced. This may be followed by message 9974.
PARAMETERS	p1 : internal Transport SAP number (hexadecimal) p2 : proposed TPDU size (hexadecimal) p3 : options (hexadecimal): a0 = TP4 for CLNS, or TP2 with alternate TP0 for CONS 00 = TP0 p4 : quality of service (hexadecimal) p5 : destination Transport selector (hexadecimal; first byte is the length) p6 : destination NSAP (hexadecimal; first byte is the length) p7 : local Transport selector, if sent in the PDU (hexadecimal; first byte is the length) p8 : local NSAP, if sent in the PDU (hexadecimal; first byte is the length)
ACTION	None.

MESSAGE	`[9973] T-DATA-Ind file`
	`[....] Buffer1 is User Data EOT=p1`

CAUSE	Trace message that is written when a Transport Data Indication is received through a Transport SAP that is being traced. This is followed by message 9974.
PARAMETERS	p1 : The end-of-TSDU flag (80 = end, 00 = partial TSDU)
ACTION	None.

MESSAGE	**`[9973] T-DATA-Req file`**
	`[....] Buffer1 is User Data EOT=p1`

CAUSE	Trace message that is written when a Transport Data Request is sent through a Transport SAP that is being traced. This is followed by message 9974.
PARAMETERS	p1 : The end-of-TSDU flag (80 = end, 00 = partial TSDU)
ACTION	None.

MESSAGE	**`[9973] T-DISCONNECT-Ind file`**
	`[....] Buffer1 is User Data Reason=p1`

CAUSE	Trace message that is written when a Transport Disconnect is received through a Transport SAP that is being traced. The Reason parameter is present only if the disconnect is not normal.
PARAMETERS	p1 : reason code (hexadecimal)
ACTION	Take action based on the reason code.

MESSAGE	**`[9973] T-DISCONNECT-Req file`**
	`[....] Buffer1 is User Data`

CAUSE	Trace message that is written when a Transport Disconnect is sent through a Transport SAP that is being traced.
ACTION	None.

MESSAGE	**`[9973] T-READY-Ind file`**
	`[....]`

CAUSE	Internal flow-control between the Transport layer and the Transport user (Session or an XTI application). The trace message is written when the Transport SAP is being traced and it is ready to accept an SPDU from the Transport user.
ACTION	None.

MESSAGE	`[9973] T-READY-Req file`
	`[....]`

CAUSE	Internal flow-control between the Transport layer and the Transport user (Session or an XTI application). The trace message is written when the Transport SAP is being traced and the Transport user indicates that it is ready to receive an SPDU from the Transport layer.
ACTION	None.

MESSAGE	`[9974] file`
	`[....] p1`

CAUSE	Trace message that accompanies message 9973 when service or user data is present in a PDU. It shows a hexadecimal representation of the beginning portion of the indicated buffer. The message "Buffer too small" indicates that format buffer is too small to show the entire contents.
PARAMETERS	p1 : hexadecimal dump of the buffer
ACTION	None.

MESSAGE	`[9976] file`
	`[....] SHcntrs=p0 p1 p2 p3 p4 LGcntrs=p5 p6 p7 p8`

CAUSE	Part of output from otsop show ent all.
PARAMETERS	p0-p4 : short entity stats counters p5-p8 : long entity stats counters
ACTION	None.

Console Messages

MESSAGE	`OTS : Cannot communicate with X.25`
CAUSE	X.25 is down.
ACTION	None.

9

Addressing

General concepts about addressing, how NSAPs are structured, and various standard addressing formats in use

Addressing

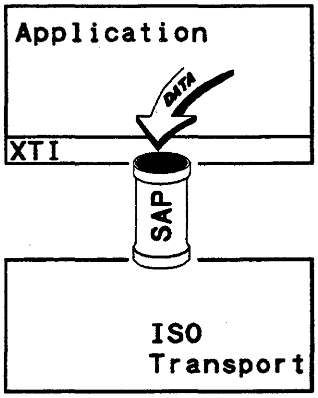
This chapter discusses general concepts about addressing, how NSAPs are structured, and various standard addressing formats in use.

General Concepts

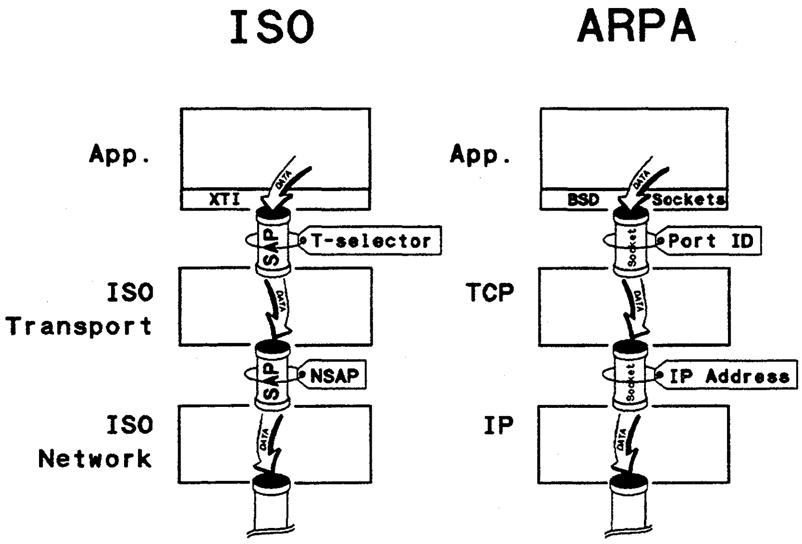
SAP - A SAP, or Service Access Point, is a "pipe" between two OSI layers that allows one layer to obtain a set of services from another layer. The services obtained vary from layer to layer, but are usually management functions, such as connection establishment and termination, and data transfer functions. For instance, a SAP between an application and the OSI Transport is a "pipe" the application uses to open connections, for example, send/receive data using the OSI Transport service.

A SAP is similar to a Berkeley socket. A BSD (Berkeley Software Distribution) socket is a "pipe" between an application and TCP or UDP used to create connections and send/receive data.

SAPs may be created between any adjacent layers in the ISO protocol suite. These SAP "pipes" can then be connected together to form a whole "conduit" through the protocol stack which allows a user to send data to, and receive data from, a remote system.



In the ARPA protocol suite, "conduits" through a protocol stack are relatively simple. They consist of a layer 4 (TCP or UDP) SAP (socket), and an IP SAP. The situation is more complex in ISO because "conduits" can extend from layer 3 up to layer 7, so several SAPs may be required to build a complete "conduit" for an ISO application.

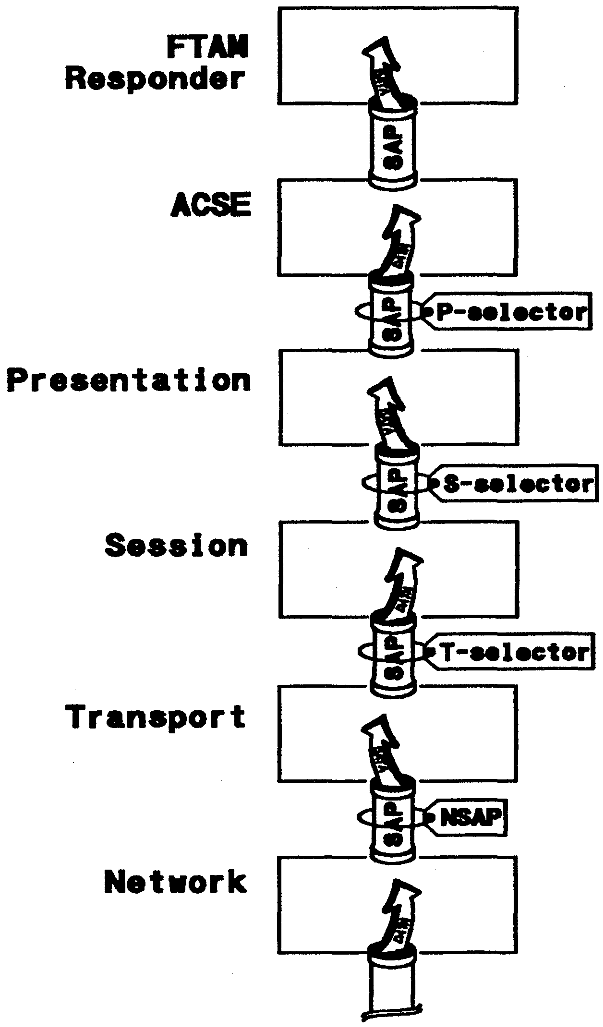


Selector - A selector is a sequence of octets (bytes) used to identify a SAP. Using the BSD socket analogy, a SAP is the socket, a selector is the two octet Port ID which is bound (using *bind()*) to the socket. When a selector identifies a specific SAP at a specific layer, it is referred to as either a P-selector, S-selector, or T-selector, so it is clear which layer the selector is associated with.

Note Because of the relationship between SAPs and selectors, the terms are sometimes used synonymously. If the term SAP (for example, PSAP, SSAP, TSAP) appears in HP documentation, it should be taken to mean the selector (that is, P-selector, S-selector, T-selector) representing the SAP.

Address - An address is a sequence of selectors, plus at least one NSAP (network service access point), which identifies an entire "conduit" through a protocol stack. For instance, in the case of FTAM, the address of its "conduit" includes the presentation, session, transport, and network layers. Its address is therefore: a P-selector, S-selector, T-selector, and one or more NSAPs. This address is commonly called a presentation address (P-address) since it defines a "conduit" whose top-most layer is presentation. More than one NSAP may be included in a P-address because the system may be reachable on more than one network (for example, a system which is accessible via both a CONS network and a CLNS network). This situation is similar to an ARPA system which is connected to more than one IP network and therefore has more than one IP address.

Applications access SAP "conduits" by using a programmatic interface. An example of a programmatic interface is X/Open's Transport Interface (XTI), which gives applications the ability to communicate with remote systems using the OSI Transport layer. The application tells the programmatic interface which "conduit" to use by passing an address to the appropriate interface procedure call. In the case of XTI, the address of the "conduit" a local application wishes to listen on is passed during the t_bind() procedure call. If the local application wishes to communicate with a remote application, it passes the remote application's address to XTI during the t_connect() procedure call.



FTAM Presentation Address

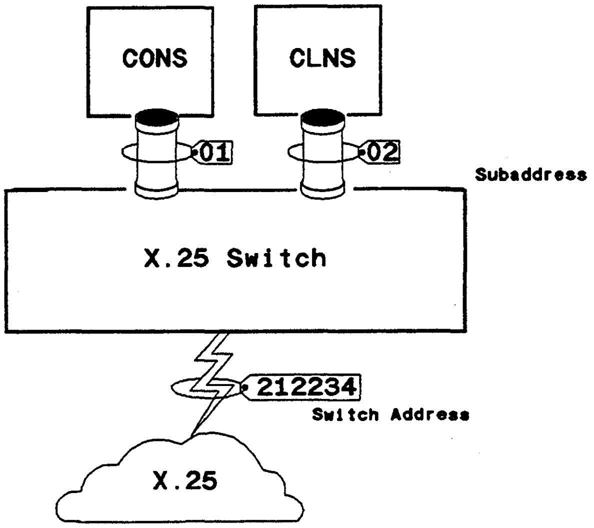
Network Layer

Unlike the ARPA protocol suite which only has a single network protocol (IP), ISO has defined two network layer services: CONS (Connection-Oriented Network Service) and CLNS (Connectionless Network Service). OTS supports CONS over the X.25 protocol, and CLNS over the X.25 and 802.3/FDDI protocols. The OTS programmatic interfaces, such as XTI, are designed to allow applications to communicate over either network service. The choice of which service (CONS or CLNS) to use is made in one of two ways. The first way is by OTS automatically examining the destination address and determining over which network service this address may best be reached. The second way to use the HP-UX *bind* command to specifically identify one service or the other.

The OSI protocol suite communicates with other systems over physical subnetworks. Examples of physical subnetwork types are X.25 and 802.3. An address as defined above is used by the OSI protocol suite to route information up from, and down to, the network layer. It is the job of the network layer to route information to destination NSAPs over the proper subnetwork. This is called network routing.

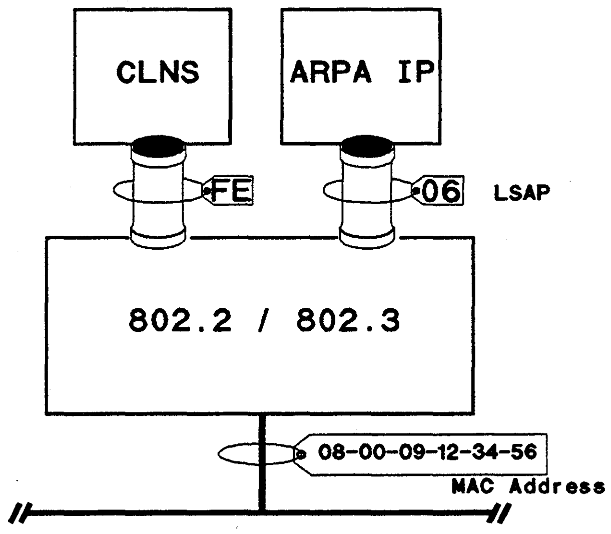
Network routing is accomplished by associating a destination NSAP with its point of attachment on a physical subnetwork, that is, the point on a physical subnetwork where that NSAP may be reached. An NSAP's point of attachment onto a subnetwork is identified by a subnetwork address.

On an X.25 subnetwork, a point of attachment is identified by an X.121 address. An X.121 address is one to 15 digits in length and has two parts: the switch address portion, and the subaddress portion. The switch address tells the X.25 protocol which switch on the subnetwork is the destination. The subaddress portion tells the switch which entity above X.25 is to receive the sent information.



In this example, two X.121 addresses have been defined: the subnetwork address for the CONS entity is 21223401, and the subnetwork address for the CLNS entity is 21223402. See the *HP 9000 Series 800 and 300 X.25 Node Manager's Guide* for a complete description of X.121 addresses and their usage.

Note Although the use of subaddresses is the recommended method to identify different X.25 users, some switches and X.25 networks do not support its use. If this is the case for your installation, see the *Installing and Administering OSI Transport Services* manual for more information.



On an 802.3 subnetwork, a point of attachment is identified by an IEEE MAC address. Instead of configured subaddress portions, the 802.3 protocol uses embedded, well known LSAP (Link Service Access Point) values for the CLNS and ARPA IP entities.

The network layer obtains its routing information in one of two ways: automatically using the ISO End System to Intermediate System Routing Protocol (ES-IS), or from statically configured tables entered by using the OTS Destination System and Routes configuration screens in *osiadmin*.

Structure of NSAP Addresses

A Network Service Access Point address (NSAP) is the ISO-defined Internet Address. It is used to identify real systems unambiguously on a network. NSAPs are used by OTS/9000 in much the same way as ARPA Internet addresses are used by TCP/IP.

There are several documents which may help you better understand this section. Among them are

- ISO 7498/AD3, Information Processing Systems - Open Systems Interconnections - Addendum to the OSI Reference Model Covering Naming and Addressing
- ISO 8348/AD2, Information Processing Systems - Data Communications - Addendum to the Network Service Definition Covering Network Layer Addressing
- ISO 9542, Information Processing Systems - Data Communications - End System to Intermediate System Routing Exchange Protocol for Use in Conjunction With the Protocol for the Provision of the Connectionless-mode Network Service [ES-IS Protocol]
- ISO 10589, Information Processing Systems - Data Communications - Intermediate System to Intermediate System Intra-Domain Routing Exchange Protocol [IS-IS Protocol]

These documents, as well as many others, are available for a small fee from

Omnicom, Inc.
115 Park Street, SE
Vienna, Va 22180-4607 USA
Telephone: (USA) 703 281-1135

Omnicom International, Ltd.
17 Park Place
Sevenage, Herts. SG1 1DU UK
Telephone: (UK) 44 438 742424

Other documents describing specific addressing formats are mentioned in their respective sections.

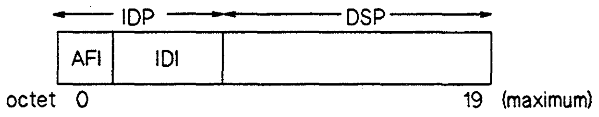
The structure for NSAP addresses discussed here has been formulated to meet the following objectives:

- Provide a framework so that HP can recommend a management strategy.
- Take into account the needs of different network types (individual or "small," sophisticated or "big," and intermediate).
- Facilitate efficient routing in private networks.
- Provide a strategy which takes advantage of dynamic routing protocols such as ES-IS and the emerging IS-IS protocols.

- Minimize the risk of re-structuring, due to present lack of universally accepted standards for NSAP structures.
- Minimize the risk of re-allocation of values, due to present lack of administrative authorities.
- Propose a method for automatic allocation of unique NSAP addresses.

The Syntax of an NSAP Address

An NSAP, as defined by ISO, has several characteristics. It may be one to twenty octets (bytes) in length. It is composed of two parts: the Initial Domain Part (IDP) and the Domain Specific Part (DSP). The DSP may also be partitioned into several fields. These fields and their sizes are defined by the authority which controls the IDP.



The Initial Domain Part

The IDP portion of an NSAP declares which national or international group owns the right to manage an NSAP space. This group is called an authority. Examples of authorities which presently manage NSAP spaces are AFNOR, ANSI and NIST. They have been given, by ISO, specific IDP values which they control and have the right to use for the distribution of NSAP addresses. As an example, NIST has been given the IDP value 470005 for use within the US GOSIP networks. Therefore, any NSAP which begins with this value is ultimately under the control of NIST. NIST also owns the right to define how the DSP portion of its NSAP space is to be formatted and used.

The IDP contains two fields. The authority and format identifier (AFI) field identifies the type of address used in the DSP. The initial domain identifier (IDI) field identifies which domain the DSP part belongs to.

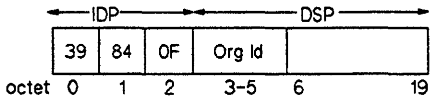
Note In this section, all numeric values are represented as hexadecimal digits unless otherwise specified.

The Domain Specific Part

The purpose of the DSP is the following:

- to allow authorities to further delegate control of NSAP addresses
- to uniquely identify a real open system unambiguously on the network
- provide information which may be used to facilitate the routing of data on concatenated subnetworks.

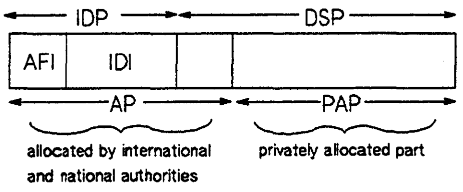
This is accomplished by dividing the DSP into a number of fields, each with its own meaning. As an example, consider the format defined by ANSI:



The AFI value of 39 indicates that the IDI portion will be a Data Country Code (DCC), and the DSP will be encoded in binary. The value 840 (padded with F) is the DCC for the United States.

ANSI has defined the first three octets of the DSP to contain an Organization Identifier (Org Id). This number, assigned by ANSI, allows other organizations to control a subset of ANSI's NSAP space. ANSI has left the rest of the DSP undefined so that the organization will have the ability to define their own address structure. For instance, if Joe's Grommet Shop petitions ANSI for an Org Id value and is assigned the value 010101 (three octets), then Joe's Grommet Shop controls the prefix 39840F010101. Any NSAP which begins with this value is under the control of Joe's Grommet Shop. Joe's Grommet Shop may also define the format for the rest of the DSP to suit their needs.

The fields of an NSAP create a hierarchy where each field further divides the NSAP space into smaller, more manageable spaces. Also, the leftmost portion of an NSAP deals with the administration of NSAP spaces, not specifically with the identification of real systems. In the above example, the leftmost part of the NSAP space is "publicly" administered, that is, administered by national and international standards bodies, whereas the right part is left up to private organizations to define and use. For convenience, these two parts will be called the administrative prefix (AP), and the privately allocated part (PAP).



This separation between what a network administrator is given and what the administrator controls is very important. It is the key to understanding how an NSAP space is managed. Also, some companies may have several different APs, or may start with one AP and have to migrate over to a different one at some point in time. Only through careful management of the PAP can a network administrator minimize the problems that this may cause.

The Privately Allocated Part

Whereas the AP is used to administer NSAP spaces, the PAP is used to define a routing strategy. The two portions, when combined together, yield a unique NSAP for a given system.

The ISO model of routing contains the following levels:

Subnetwork - A subnetwork is an autonomous collection of equipment and medias used to interconnect systems. Examples of types of subnetworks are: IEEE 802.3 LANs and X.25 PDNs.

Area - An area is a group of end systems and intermediate systems interconnected by one or more subnetworks. They have been grouped together by way of an autonomous routing mechanism. An autonomous routing mechanism is either a set of statically configured intermediate systems, or a set of intermediate systems which support a dynamic routing protocol, such as the ISO IS-IS protocol.

Routing Domain - A Routing Domain is also a group of end systems and intermediate systems interconnected by one or more subnetworks. They have been grouped together by way of some routing or security policy defined by the network administrator. Typically, a number of areas are grouped together to form a routing domain.

An area may be composed of one or more subnetworks. The criteria for grouping subnetworks into areas can be based upon topology, reachability, and network traffic. The criteria for grouping areas into routing domains is based upon policy. For instance, a company might want to create an area for their factory, and area for their product design lab and put them in the same routing domain. However, the company may choose to create a separate routing domain for their accounting and payroll departments.

To facilitate routing strategy, a network administrator will often define the PAP to contain one or more of the following fields:

Routing Domain Identifier
Area Identifier
Subnetwork Identifier
End System Identifier

The order which they are listed here is the order, left to right, in which they would be defined in a PAP.

Other fields which often find their way into the PAP are:

Version Number or *DSP Format Identifier* - The purpose of these fields is to allow a network administrator to redefine the PAP structure at a later point in time, or allow the network administrator to define several PAP formats. Each format would be identified by a different value in the Version Number or DSP Format Identifier field.

Reserved field - This is simply a portion of the PAP which is not in use, but has been reserved in case it is needed some time in the future. If the network administrator defines a PAP with a reserved field, a default value (usually all 00s) should be defined which the reserved field is always set to.

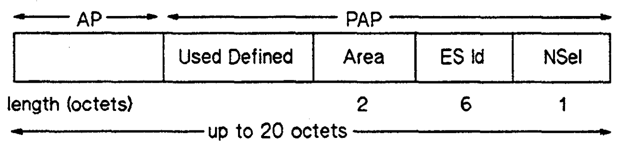
NSAP selector - A one octet field at the rightmost portion of the PAP. It identifies the entity attached to the network layer. In OTS the entity is always the transport protocol. HP recommends that the NSAP selector be set to 01.

The section on current allocation formats, and the section on recommended PAP structures show examples of how these fields are used to manage NSAP spaces and routing topologies.

ISO IS-IS Routing protocol - ISO is in the process of defining a protocol standard for a dynamic Intermediate System to Intermediate System routing protocol, known as the IS-IS protocol. This protocol, when complete, will allow Intermediate Systems (the ISO term for network layer routers) to automatically route traffic between each other across concatenated subnetworks and areas. It will not, however, automate routing traffic between routing domains. This will still require static configuration of route information.

Though the protocol is still in development, ISO has defined a format for the rightmost portion of the DSP to be used by the IS-IS protocol for its automated routing. Network administrators should use this format whenever possible so that it is easier to support the IS-IS protocol once it becomes available.

The format defined by ISO is the following:



Area ID	A two octet field containing a unique ID for the area on which the NSAP resides.
ES ID	A six octet value, unique within the Area, which identifies the end system.
NSel	A one octet value. The recommended value is 01.

This format is commonly called the 2/6/1 structure because of the sizes of the three defined fields.

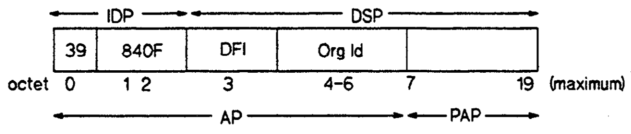
This is the format that NSAP addresses should obey in order to get the most use out of the automated routing protocols. See the section "A Recommended PAP Structure" later in this chapter for a possible PAP structure which network administrators may use to define NSAP formats. This recommended structure is compatible with the IS-IS protocol format.

Current NSAP Allocation Formats

This section shows a number of examples for existing NSAP allocation formats. Other standards organizations may have additional formats which customers may want to use. Customers can contact standards organizations within their home country to obtain more information.

Note HP recommends that customers use the binary format for NSAPs. Therefore, only this format is discussed here.

ANSI Format

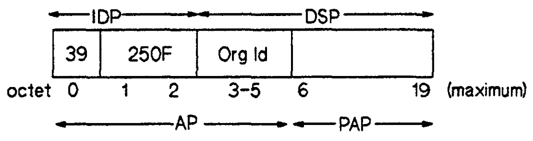


ANSI is the standards body which controls the United States Data Country Code (DCC) 840. They have defined the first three octets of the DSP to be an Organization Identifier, leaving the other 14 octets (the PAP) to be defined by each controlling organization. ANSI may be reached at the following address:

American National Standards Institute
1430 Broadway
New York, New York, 10018, USA
Telephone: (USA) 212 642-4932

Note ANSI defines a recommended PAP structure for their NSAP space. It's syntax may be obtained by writing to the above address.

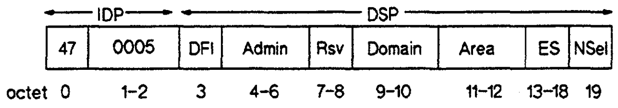
AFNOR Format



AFNOR (Association Francaise de NORmalisation) is the standards body which controls the French Data Country Code (DCC) 250. Like ANSI, they have defined the first three octets of the DSP to be an Organization Identifier, leaving the other 14 octets (the PAP) to be defined by each controlling organization. AFNOR describes their addressing format in its document, *X 60-000*. AFNOR may be reached at the following address:

AFNOR
Tour Europe - Cedex 7
F-92049 PARIS LA DEFENSE (FRANCE)
Telephone: +33 1 42 91 55 55

United States GOSIP



DFI	DSP Format Identifier. This is used to specify the structure, semantics and administration requirements for the remainder of the DSP. Currently, only one DSP Format exists. This field is similar in function to a Version field.
Admin	This field functions in much the same way as the Org ID field in the ANSI format. It is an Administration authority identifier. The values for this field are under the control of NIST.
Rsv	Two octets are reserved for future expansion.
Domain	The Routing Domain this NSAP resides in. Routing Domain values are determined by the authority identified by the Admin field.
Area	The Area number this NSAP resides in. Area values are determined by the authority identified by the Admin field.

ES	A number, unique within the area, which identifies the end system. End System identifiers are determined by the network administrator.
NSel	This field identifies the Network Service user. U.S. GOSIP recommends this value be set to 01 for the Transport protocol.

The United States GOSIP format, whose administration authority is NIST, is to be used by United States Government agencies for their OSI networks. NIST will also assign NSAPs from their NSAP space to non-government organizations upon request.

This format is a good example of how the various PAP fields may be used to create a well defined NSAP space which can evolve over time. U.S. GOSIP has taken a domain/area approach to routing and has defined the entire structure of the DSP. Because of this, they have also added some fields which will allow their structure to change over time, namely the DFI and Rsv fields.

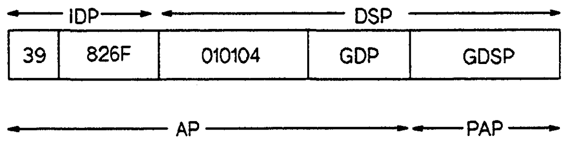
To obtain more information about the U.S. GOSIP NSAP structure, contact:

Telecommunications Customer Requirements Office
U. S. General Services Administration
IRMS
Office of Telecommunications Services
18th & F Sts. N.W.
Washington, D. C., 20405

A more complete discussion of this addressing format may be found in *U.S. Government OSI Profile Specification Version 2.0*

This document is produced by the National Institute of Standards and Technology, USA (NIST). It should be available through Omnicom or NIST.

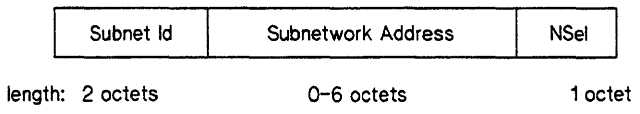
United Kingdom GOSIP



GDP	Government Domain Part (one octet)
GDSP	Government Domain Specific Part (up to 11 octets)

The United Kingdom GOSIP program, like ANSI, has chosen the Data Country Code format for their IDP. GDP values are provided by the Central Computer and

Telecommunications Agency (CCTA), and functions in a similar fashion as US GOSIP's Admin field. The GDSP is undefined and is left up to the controlling organization (identified by the GDP) as to its structure. U.K. GOSIP recommends that the following fields be defined as the rightmost portion of the GDSP:



The Subnetwork Identifier may identify a physical subnetwork, or an area. The Subnetwork Address field may contain a real subnetwork address, such as a MAC or X.121 address, or a virtual end system identifier which uniquely identifies the end system for the specified Subnet ID. HP recommends that the NSel field be set to 01.

A more complete discussion of this addressing format may be found in:

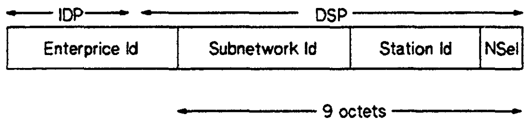
U.K. Government OSI Profile Specification Version 3.1

Procedure for Obtaining a U.K. Government Domain Part (GDP) under the ISO DCC Addressing Scheme.

These documents are produced by the Central Computer and Telecommunications Agency, UK. They should be available through Omnicom International, Ltd. or CCTA.

GOSIP Project Office
CCTA
Riverwalk House
157/161 Millbank
LONDON SW1P 4RT UK

MAP/TOP 3.0



Enterprise ID	The AP portion of the NSAP. It includes the AFI, IDI and, if necessary, an Org ID.
Subnetwork ID	A portion which uniquely identifies the Subnetwork on the Enterprise.

Station ID	A portion which uniquely identifies an end system on the subnetwork
NSel	The field which identifies the entity above the network layer.

Unlike ANSI and AFNOR, which defined an AP and left the PAP undefined, MAP/TOP has defined a general format for the PAP portion. Their PAP may be used with any valid AP.

MAP/TOP has decided to use subnetworks as their level of routing rather than areas. This is more in line with present ARPA routing procedures than the upcoming ISO routing protocols. MAP/TOP does recommend that the customary 2/6/1 field lengths be used for the PAP portion to allow easy migration to area-based routing in the future.

A more complete discussion of this addressing format may be found in *Manufacturing Automation Protocol Specification, Version 3.0,* available from the MAP/TOP users group:

North American MAP/TOP Users Group
ITRC
P.O. Box 1157
Ann Arbor, MI 48106 USA

Building Temporary Administrative Prefixes

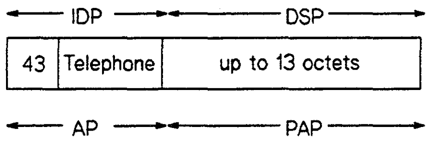
In many cases, the examples above will not fit the needs of many users. This is because they are not government organizations, the company does not reside in a country which has an NSAP authority, or the company does not wish to pay the registration fees required to obtain an NSAP space.

In cases like these, a network administrator can create a temporary Administrative Prefix which their company may use as an interim solution until a more permanent AP becomes available. The network administrator must be sure to create a PAP format which can easily be migrated to the permanent NSAP space, once it becomes available.

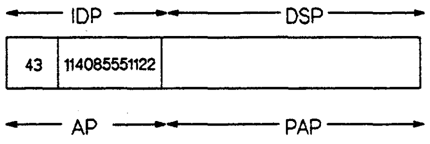
The first two methods described create a unique NSAP space, and the NSAP addresses may be used on open networks (that is, networks connected to a public network). The last method, Local Format, does not guarantee a unique NSAP space and should only be used when connectivity to OSI networks outside the customer's network is not required.

Use an International Telephone Number

ISO allows NSAP spaces to be built which have an International Telephone Number as their IDI value. This is known as the E.163 format. The AFI code to use is 43. The telephone number is encoded as Binary Coded Decimal (BCD) into the IDI. Up to 12 digits may be used. Any unused digits should be set to F (hexadecimal). The DSP portion, and therefore the PAP, may be up to 13 octets in length.

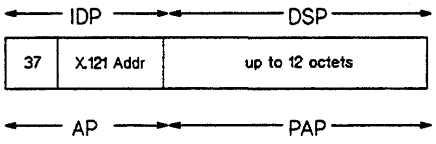


As an example, suppose Joe's Grommet Shop decides not to obtain an ANSI Org Id at this time. Instead, they use the E.163 format. The network administrator decides to use the International Telephone Number of their customer order department for the IDI value.

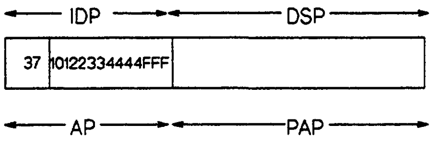


Use an X.121 Address

Another option is to use an X.121 address as the IDI value. The AFI for this format is 37. The IDI is up to 14 BCD encoded digits, with any extras padded with F. The DSP, and therefore the PAP, may be up to 12 octets in length. Note that the X.121 address is only used to uniquely identify the customer's AP. It does not have any routing significance in this example.

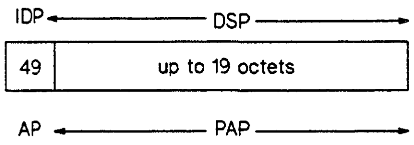


For instance, Joe's Grommet shop decides to use one of its public X.121 address for the IDI value.



Use the ISO Local Format

ISO defines a format known as the Local format. HP recommends that you do not use this format because it can result in non-unique NSAP values. They should only be used for pilot networks or for diagnostic purposes on networks which are isolated, that is, not connected to any public networks.

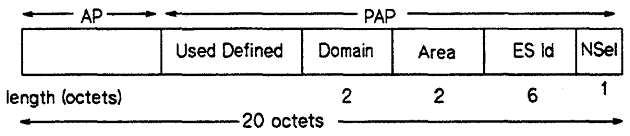


When using Local Format it is especially important to create and manage the PAP portion in a way that can be easily migrated to another AP in the future.

A Recommended PAP Structure

Here is an example of a PAP structure which may either be used with a current allocation format which allows the PAP to be defined (such as ANSI), or with a temporary AP.

The last portion of the PAP should have the following structure:



User Defined	A unique, user defined number.
Domain ID	A two octet field containing a unique ID for the routing domain on which the NSAP resides.
Area ID	A two octet field containing a unique ID for the area on which the NSAP resides.
ES ID	A six octet value, unique within the area, which identifies the end system. Possible values which may go here are: If the NSAP is to be used over 802.3, the MAC address may be used. If the NSAP is to be used over X.25, and the X.121 address space is 12 digits or less, the X.121 address may be used. The X.121 address is entered as BCD digits, two digits per octet. If the X.121 address has an odd length, pad the last octet with F.
NSel	A one octet value. The recommended value for OTS is 01.

The above PAP allocation follows the structure used by the IS-IS protocol. It allows areas to be created which contain multiple subnetworks.

The User Defined portion may be comprised of one or more of the following fields:

- A version or DSP format identifier. This allows the format of the PAP to be redefined at some later point in time. These fields are usually two octets in length, but any size may be chosen. For most NSAP spaces, one octet will often be sufficient.

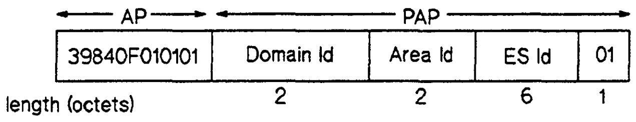
- A reserved field. It is often a good idea to reserve a few octets of space for future use.

If any of these fields are used, they should be defined in the order, left to right in the PAP, as they are listed above.

Network Identifiers (Network ID)

Because of the hierarchical definition of NSAP formats, a prefix portion of an NSAP may be used to identify a group of systems which reside in the same routing domain, the same area, or the same subnetwork. This prefix portion is called an "NSAP Prefix," or a "Network ID."

As an example, suppose Joe's Grommet Shop, whose AP value is 39840F010101, decides to use the Recommendation 1 format for its PAP.



It has two areas in routing domain 0000 which are assigned Area ID values: 0001 and 0002. The corresponding Network IDs for these areas are: 39840F01010100000001 and 39840F01010100000002.

Network IDs may be used in OTS to simplify the configuration of routing information. Rather than configuring a route to every remote system, OTS allows network administrators to configure a route to a distant routing domain, area, or subnetwork by using its Network ID. It also allows local subnetworks to have their Network IDs configured to help the routing protocols with local network traffic.

General Recommendations for NSAP Addresses

Here is a list of recommendations by HP for the allocation of NSAPs for use with OTS:

- Use an Administrative Prefix obtained from a national or international authority if at all possible. An authority does not have to reside in the same country as the petitioning organization (for example, a company in Spain could petition AFNOR for an Organization Identifier).
- If the Local Format is to be used, the network must be completely isolated from public network traffic. If a customer needs a temporary AP, they should use a International Telephone Number, or the X.121 format.
- Use the binary AFI value whenever possible.
- If using a temporary AP, limit the length of the PAP to 14 octets or less. This will allow the PAP to fit into the space allowed by a national or international authority's AP in the future.
- If using an AP allocated by an international or national authority, define all 20 octets of the NSAP, even if this means allocating a reserved field.
- Define and manage the PAP address structure independent of the AP. This will make managing multiple AP values, or migrating between AP values easier. This especially includes the following fields:

  Area ID
  Routing Domain ID
  Subnet ID

  All the above fields should be unique regardless of AP value, even if the authorities administering the APs have defined the full DSP. For instance, if a network administrator is managing a MAP network and a US GOSIP network, the values used in the Subnet ID fields for the MAP network NSAP addresses should not be reused in the Area ID values in the US GOSIP NSAP addresses.
- All fields should be defined to fall on octet boundaries, that is, they should always contain an even number of hexadecimal digits.
- To be compatible with the ISO IS-IS Routing Protocol, the last three fields of the PAP should conform to the 2/6/1 format.

Non-standard NSAP Formats for Use in OTS

If the CONS network being used supports the 1984 X.25 Extended Address facility, the NSAPs configured should conform to the above recommendations.

If the CONS network being used is version 1980, or does not support the Extended Address facility, use the X.121 address of the system's attachment point onto the X.25 network in place of an NSAP.

For subnetworks which use the Null Subset of CLNP, the subnetwork address, either MAC or X.121, of the system's point of attachment onto the subnetwork should be used in place of an NSAP.

Glossary

A

ACSE/Presentation and ROSE Interface See **APRI**

Advanced Research Projects Agency See **ARPA**

ANSI The American National Standards Institute that publishes standards for use by national industries.

API A set of functions enabling an application program to interact with and control network operations and resources.

application program interface See **API**

APRI An application program interface to the OSI presentation.

argument The part of a command line that identifies what (file, directory, etc.) is to be acted upon.

ARPA The Advanced Research Projects Agency. A U.S. government research agency that was instrumental in developing and using the original ARPA services on the ARPANET.

B

backbone The principal network segment to which all **nodes** are connected, or to which other segments are connected.

BAS This subset is used with basic X.400 application and Session version 1.

basic activity subset See **BAS**

basic combined subset See **BCS**

basic synchronized subset See **BSS**

BCS This subset is used with basic FTAM applications and Session version 2.

boot To start up your system, loading it into the computer memory.

bridge A device that connects different LANs.

BSS This subset is used with advanced user application and advanced FTAM usage.

bypass A mechanism to avoid sending data to a faulty device or portion of the network.

C

CCITT Consultative Committee for International Telegraphy and Telephony. An international organization of communication carriers.

CLNP subset Indicates the LAN subnetwork information.

CLNS Connectionless-oriented network services.

common management information service (CMIS) The interface for development of network management applications.

CONS Connection-oriented network services.

command A word or phrase that you type at the system prompt to carry out an action when you press the ENTER key.

connection-oriented network services See CONS

connectionless-oriented network services See CLNS

Consultative Committee for International Telegraphy and Telephony See CCITT

configure To set up your computer system so that the computer and all peripheral devices can work together. If the computer is part of a network, this includes loading the appropriate software and establishing the necessary connections.

configuration The arrangement of a computer system or network as defined by the nature, number, and the chief characteristics of its functional units. More specifically, the term configuration may refer to a hardware configuration or a software configuration.

cug Closed user group. An X.25 user facility that allows a predetermined group of users to contact and be contacted by members of the group alone.

D

daemon A software process that runs continuously and provides services on request.

distributed system A computer system in which computing, storage, and other resources are dispersed throughout several or many locations.

E

Ethernet A 10 Mb/s LAN, developed by Digital Equipment Corporation, Intel, and Xerox Corporation, upon which the **IEEE 802.3 network** is based.

F

FDDI A specification for a fiber-optic ring network featuring a link speed of 100 Mb/s and fault tolerant capabilities.

fiber distributed data interface See FDDI

fileset Describes the logical, defined set of files on an update or installation tape.

file transfer, access, and management See FTAM

file transfer protocol See FTP

FTAM Provides the capability to manipulate data files locally and at remote nodes.

FTP The file transfer protocol that is traditionally used in ARPA networks. The **ftp** command uses the FTP protocol.

G

gateway A node that connects two or more networks together and routes packets between those networks.

GOSSIP Government OSI Protocol. An OSI-based network protocol used by governments (for example, the United States and United Kingdom).

H

heterogeneous network A network composed of dissimilar host computers, such as those of different manufacturers. See **homogeneous network** for contrast.

homogeneous network A network composed of similar host computers, such as those of one model or one manufacturer. See **heterogeneous network** for contrast.

I

IEEE The Institute of Electrical and Electronics Engineers. A national association, whose activities include publishing standards applicable to various electronic technologies.

IEEE 802.3 network A 10-megabit-per-second LAN, described by the ANSI/IEEE 802.3 Standard for Local Area Networks, that uses a CSMA/CD network access method.

IOP Interoperability procedures used to verify that nodes can communicate over the network.

ISO The International Standards Organization that created a network model identifying the seven commonly-used protocol levels for networking.

K

kernel The part of the HP-UX operating system that is an executable piece of code responsible for managing the computer's resources.

L

LAN A data communications system that allows a number of independent devices to communicate with each other.

LLC (logical link control) The ANSI FDDI standard that provides a common protocol between the MAC function in the data link layer and the network layer.

Local Area Network See LAN

local network The network to which a **node** is directly attached.

local network ID Some initial set of digits of the NSAP that form a prefix for all systems reachable over this subnetwork.

M

MAC The ANSI FDDI standard that defines the data link layer function responsible for the scheduling, routing and delivery of frames on and off the FDDI ring.

manufacturing message specification See MMS

media access control See MAC

MMS Provides the capability to control and coordinate programmatic factory floor devices involved in manufacturing.

N

network address See NSAP

network administrator An individual responsible for network administration, for example, organizing network domains and issuing node names.

network architecture The set of principles, including the organization of functions and the description of data formats and procedures, that governs the design and implementation of a user-application network.

node Any point in a network where services are provided or communications channels are interconnected. A node could be a workstation or a server processor.

NSAP A unique value that defines a system's address for use when establishing network connections among various systems.

O

open system interconnection See OSI

OSI Open System Interconnection reference model defined by the International Standards Organization (ISO). It establishes a data communication architectural model for networks.

OTS HP's term for the OSI transport services.

P

packet A sequence of binary digits that is transmitted as a unit in a computer network. A packet usually contains control information plus data.

PID A unique identification number assigned to all processes by the operating system.

port A software access point for data entry or exit to a network controller.

process identifier See **PID**

protocol A specification for transferring information between computers on a network.

R

redundancy Duplication of service. Networks can provide redundancy to increase the probability that communications can continue despite various failures.

remote Not directly connected or processed at another location.

routing node A **node** that is able to transmit **packets** between similar networks. A node that transmits packets between dissimilar networks is called a **gateway**.

S

SAP Service access points between network layers.

selector A sequence of octets (bytes) used to identify a SAP; referred to as p-selector, s-selector, and t-selector to identify the OSI layer association.

service access point See **SAP**

SAS A station in an FDDI network that connects to only one of the two FDDI network rings. An SAS must attach to the network through a concentrator.

session interface An application program interface to the OSI session layer.

single attachment station See **SAS**

SMT The ANSI FDDI standard which manages connections with the ring as well as station and ring configuration.

station management See **SMT**

subnetwork A group of computers that are a part of a larger network and whose IP address includes a subnetwork number.

system administrator The person who oversees system maintenance and computer operation.

T

topology The physical and logical geometry governing placement of **nodes** in a computer network. Also, the layout of the transmission medium for a network.

X

X/open transport interface See XTI

XTI An application program interface to the OSI transport layer.

X.21 Defines the interface between a computer and a public data network where the access to the network is made over synchronous digital lines.

X.25 Defines the interface between a computer and a packet switching network.

X.400 The interface for electronic messaging applications over a network.

X.500 A distributed directory interface allowing different vendors to store and access information on different systems.

Index

The HP OSI Series

For Information On:	**Read:**
HP-UX (HP 9000, Series 800, HP 9000, Series 400, and HP 9000, Series 300)	*HP-UX Reference* (3 volumes) (PN:09000-90013) *Shells and Miscellaneous Tools, HP-UX Concepts and Tutorials* (PN: 97089-90062)
X.25/9000	*Installing and Administering X.25/9000* (PN: 36940-90004) and *Troubleshooting X.25* (PN: 36940-90005)
LAN/9000	*Installing and Administering LAN/9000 Series 300* (PN: 98190-90015) and *Installing and Administering LAN/9000 Series 800* (PN: 98194-90016)
FDDI/9000	*Installing and Administering FDDI/9000 Software* (PN: J2156-61001)
OTS/9000	*Installing and Administering OSI Transport Services* (PN: 320-69-60003)
Session API	*Session Access Programmer's Guide* (PN: 32069-60004)
XTI API	*HP-UX/9000 XTI Programmer's Guide* (PN: 32069-60002)
APRI	*ACSE/Presentation and ROSE Interface Programmer's Guide* (PN: 32069-60005)
FTAM/9000	*HP FTAM/9000 Reference Manual* (PN: B1033-60500), *HP FTAM/9000 User's Guide* (PN: B1033-60520), *HP FTAM/9000 Programmer's Guide* (PN: B1033-60510), and *Installing and Administering HP FTAM/9000* (PN:)
MMS/9000	*HP MMS/9000 Reference Manual* (PN: 32019-60500), *HP MMS/9000 Programmer's Guide* (32019-60510), and *Installing and Administering HP MMS/9000* (PN:)
HP X.400	*Installing and Configuring HP X.400* (PN: 32034-90005), *HP X.400/HP Desk Node Administrator's Guide* (PN: 32055-90001), *Using HP DeskManager Connected to X.400* (PN: 32055-90002), *Managing HP X.400* (PN: 32034-90006), *Using HP X.400 with Elm and Mailx* (PN: 32034-90007), *HP X.400 High-Level API* (PN: 32034-90008)

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 2	July 1991
Edition 3	March 1992

List of Effective Pages

The List of Effective Pages gives the date of the current edition and of any pages changed in updates to that edition. Within the manual, any page changed since the last edition is indicated by printing the date the changes were made on the bottom of the page. Changes are marked with a vertical bar in the margin. If an update is incorporated when an edition is reprinted, these bars are removed but the dates remain. No information is incorporated into a reprinting unless it appears as a prior update.

Pages	**Effective Date**
All	March 1992

HEWLETT PACKARD

Customer Order No.
32069-60001

Manufacturing No.
32069-90009
Mfg. number is for HP internal use only

32069-90009

HEWLETT
PACKARD

Customer Order No.
32069-60001

Manufacturing No.
32069-90009
Mfg. number is for HP internal use only

32069-90009