# Bull DPX/20

## OSI Diagnostic Interactive Toolkit (ODIT)

AIX

ORDER REFERENCE
**86 A2 39WG 03**

# Bull DPX/20

## OSI Diagnostic Interactive Toolkit (ODIT)

AIX

**Software**

**June 1996**

**Bull Electronics Angers S.A.**
**CEDOC**
Atelier de Reprographie
331 Avenue Patton
49004 ANGERS CEDEX 01
FRANCE

ORDER REFERENCE
**86 A2 39WG 03**

Suggestions and criticisms concerning the form, content, and presentation of this book are invited. A form is provided at the end of this book for this purpose.

## Trademarks and Acknowledgements

# About This Book

The OSI Diagnostic Interactive Toolkit (ODIT) User's Guide supplies users with the following information:

- An overview of ODIT capabilities,
- Guidelines on using the OSI Diagnostic Toolkit,
- ODIT installation procedure using the System Configurator,
- A description of the ODIT menu structure,
- An overview of OSI stack information,
- A description of API trace management,
- Management of OSI stack internal traces,
- Analysis of OSI protocol.

## Audience

This manual addresses an audience of experienced UNIX system administrators requiring first level assistance to debug problems occurring during OSI stack processing. Those requiring detailed information on UNIX operating systems and the OSI stack should refer to the Bibliography on page B-1, for further reading.

## Operating System Level

This document is at Revision 03 level applying to AIX Version 4.1.

Migration from an earlier version of AIX is detailed in the *OSI Communications Porting Guide* .

## Supported Environment

All information in this manual pertains to the DPX/20 family of single- and multi-processor systems.

## Document Overview

This manual is structured in nine chapters, two appendices, a glossary and a general index.

| | |
|---|---|
| **Chapter 1** | **Introducing ODIT**<br>Description of contents and capabilities of the OSI Diagnostic Toolkit. |
| **Chapter 2** | **Using the OSI Diagnostic Toolkit**<br>Description of the OSI stack trace architecture and guidelines on using the OSI Diagnostic Toolkit. |
| **Chapter 3** | **ODIT Installation**<br>Installation of ODIT with the System Configurator. |
| **Chapter 4** | **Accessing ODIT**<br>Menu structure and quick guide. |
| **Chapter 5** | **OSI Stack Information**<br>Overview of OSI Stack information. |
| **Chapter 6** | **API Trace Management**<br>API trace configuration and analysis. |
| **Chapter 7** | **OSI Stack Layers Trace Management**<br>Applying trace management procedures using ODIT. |
| **Chapter 8** | **OSI Protocol Analyzer**<br>Analyzing OSI protocol with ODIT. |
| **Chapter 9** | **OSI Stack Diagnostic Toolkit**<br>Diagnostic and maintenance tools not integrated under SMIT. |
| **Appendix A** | **Trace Report**<br>Detailed description of ODIT trace report. |
| **Appendix B** | **Bibliography**<br>Further source information. |
| **Glossary** | Alphabetical list of terms and abbreviations used in this manual. |
| **Index** | **General index** |

## Conventions used in this document

The following typographic conventions are used in this document:

| | |
|---|---|
| **Bold** | Bold characters are used to highlight key words, commands and subroutines. |
| *Italic* | Italic characters represent file names and user supplied values. |
| `Courier` | Courier characters are used in examples and for user commands entered on the terminal keyboard. |
| Fixed pitch | Fixed pitch characters are used to present system display information. |
| Display fields | Individual fields on a display screen are outlined and presented sequentially. |
| # | Numeric field. |
| X | Hexadecimal field. |
| ***Mandatory** | The names of fields a user must complete are presented in bold type and with an asterisk (*) to the left. |

## References to Standards

Applicable Standards are listed in the Bibliography on page B-1.

## Bibliographical References

Reference documents, cited in text, are listed in the Bibliography on page B-1.

## Terminology

The term “machine” is used to indicate the proprietary hardware, in this case the DPX/20 family of single- and multi-processors.

The term “Operating System” is used to indicate the proprietary operating system software, in this case AIX.

The term “ODIT” is used to indicated the OSI diagnostic tools having a SMIT user interface.

## Revision 03 Modifications

This document has been updated as follows:

- **trmem** command is now embedded in **trmad**.
- ODIT Quick Guide in ’Accessing ODIT’ chapter has been condensed.

# Table of Contents

# Chapter 1. Introducing ODIT

This section introduces the OSI Diagnostic Interactive Toolkit.

## Purpose

ODIT is a set of tools and commands dedicated to solving possible problems that may occur during OSI network processing. The object of this document is to give users an overview of the OSI stack architecture and to describe the type of traces obtained and how to use them in order to identify and solve a possible problem.

## Contents

The OSI Diagnostic Interactive Toolkit is comprised of a certain number of tools listed below.

However, the term **ODIT** is generally used throughout this manual to designate those tools having a SMIT interface, namely **osiinfo**, **ositrace**, **osiprot**, **ositrcrpt**, **ositrcstart** and **ositrcstop**. They are presented with an **asterix *** in the list below and are not detailed as commands in this document.
The ODIT menu structure under SMIT is described in “Accessing ODIT” on page 4-1.This section provides cross-references to individual menu function descriptions.

The other tools, i.e. **trmad, pmaderror, lookx25, osisnap and osiping** are described individually in “OSI Stack Diagnostic Toolkit” on page 9-1.

- ***osiinfo**, which displays OSI stack information,
- ***ositrace**, which manages OSI stack traces as well as XAP, ROSE and Session API traces,
- ***ositrcstart**, used to start the trace on an API,
- ***ositrcstop**, used to stop the trace on an API,
- ***ositrcrpt**, used to analyze traces according to the grammar delivered with the package,
- ***osiprot**, which manages OSI stack protocol traces,
- **trmad**, which provides direct access to the OSI stack trace and internal trace, and memory status.
- **pmaderror**, which provides a clear explanation on error codes,
- **lookx25**, which manages x25 boards and connections,
- **osisnap**, which gathers OSI stack information if a serious problem occurs,
- **osiping,** which sends a Transport Protocol Data Unit (TPDU) to obtain a response from a host or a gateway.

## ODIT Capabilities

ODIT can operate in four different modes:

- OSI Stack information,
- OSI API trace management,
- OSI layers trace management,
- OSI Protocol Analyzer.

## OSI Stack Information

The following information can be presented:

- LPP installed,
- Status of loaded driver,
- Status of OSI daemons,
- Statistics on OSI layers,
- Alarm.

## OSI API Trace Management

The following APIs can be traced according to selected trace levels

- XAP API,
- ROSE API,
- Session API.

## OSI Layers Trace Management

Using SMIT, the user can handle OSI Stack information concerning the OSI driver and OSI layer processing. The major actions that can be performed are as follows:

- List the OSI stack trace configuration,
- Reset the OSI layer trace configuration,
- Tune the general trace configuration,
- Reset the trace buffer,
- Perform trace analysis,
- Save a trace to a file.

## OSI Protocol Analyzer

Using SMIT, the user can run a trace on all the PDUs entering or exiting the OSI stack. The OSI Protocol Analyzer can be used to decode these PDUs. It offers the following four functions:

- Reset the protocol analyzer,
- Tune the protocol analyzer,
- Run the protocol analyzer,
- Analyze the traces produced.

# Chapter 2. Using the OSI Diagnostic Toolkit

This section provides an illustrated description of the OSI stack trace architecture and guidelines for using the OSI Diagnostic Toolkit according to the user's needs.

## Understanding Error Reporting Mechanisms

### OSI Stack Error Reporting

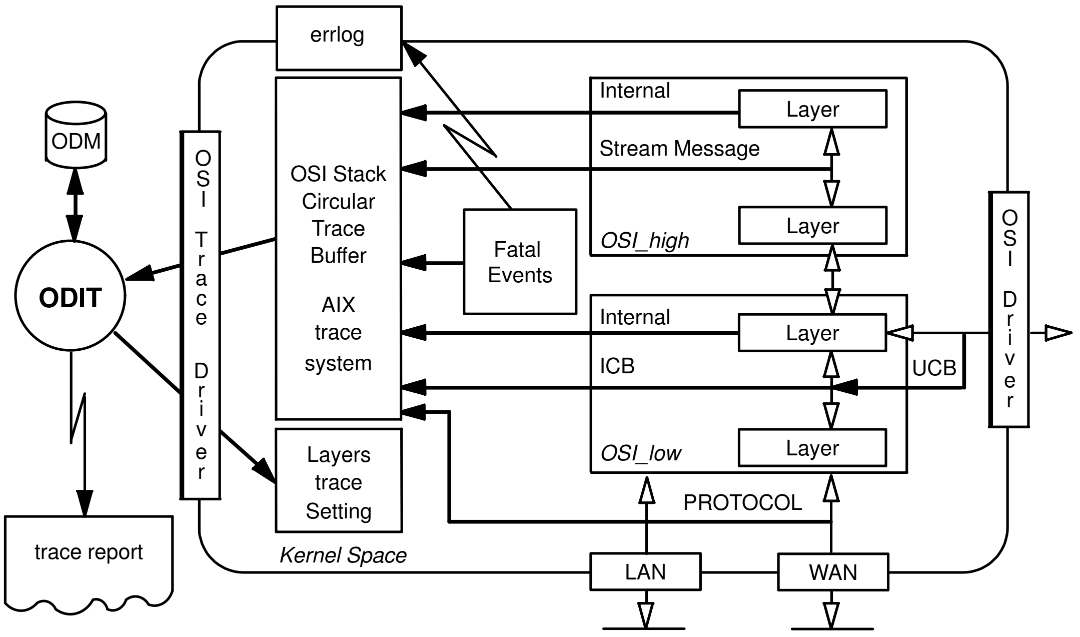


Figure 1. OSI kernel trace overview

When the user modifies a trace parameter using ODIT, the ODM base is updated accordingly and the information concerning the modified parameter is sent to the OSI stack through the OSI trace driver and taken into account immediately. See Figure 1 above. The trace layer setting is preserved during stack activation/de-activation or system reboot.

A circular trace buffer is used to collect all the OSI stack traces. All traced elements are time stamped and the tracing pid is also recorded. The various traces stored are as follows:

| | |
|---|---|
| Protocol traces | these consist of the traces caught at the level of X25.3, MAC or CLNP loopback layers for the PDUs entering or exiting the OSI stack. PDU traces can be truncated to a specified length. |
| ICB traces | ICBs are messages exchanged by the components of the `osi_low` driver. Using ODIT, these ICBs can be traced according to their origin or to their target layer. ICB traces can be truncated to a specified length. |
| UCB traces | UCBs are messages exchanged by applications such as TPAD-HPAD with the osi_low driver (X25.3 access method, administration layer or proprietary COTP access). UCB traces can be truncated to a specified length. |
| Internal traces | these are developer traces used to trace such things as errors and warnings generated, accesses to internal code functions, access to local data, calls to system resources and automata status. The traces and trace levels can be selected by the user. |

Stream message traces

Using ODIT, streams messages exchanged in the upper layers can be traced according to their origin or to their target layer. These traces can be truncated to a specified length.

Fatal event traces

A single fatal event can be traced (see section “Alarm Condition” on page 2-4). In addition to OSI stack and AIX traces, a record is also traced in *errlog* with the message “OSI Driver is in alarm”.

For specific needs, the AIX system trace can be used to produce OSI stack traces using Hook ID 900 and X25 traces using Hook ID 530. In such cases, the decoding analysis must be performed by a Technical Support stack specialist.

ODIT and **trmad** allow to configure trace parameters for each class of trace and to display formatted trace records. The formatted trace can be saved in the directory defined by the OSI_TRACE_DIR variable (or by default in */var/tmp/osi*).

A scan mode allows to display information as it is being traced. If this mode is not enabled, the contents of the circular trace buffer are displayed.

Another mechanism for error reporting is based on error and diagnostic codes which are exchanged by the stack layers. When a problem occurs, the error or diagnostic code is propagated from the source layers through the layers above and to the application. See section “Error and Diagnostic Codes Analysis” on page 2-3. Fatal events also produce similar codes but they are not transmitted to the APIs.

## API Error Reporting

This section is mainly intended for API users.

The figure below illustrates the error reporting mechanism for XAP, ROSE and Session APIs.

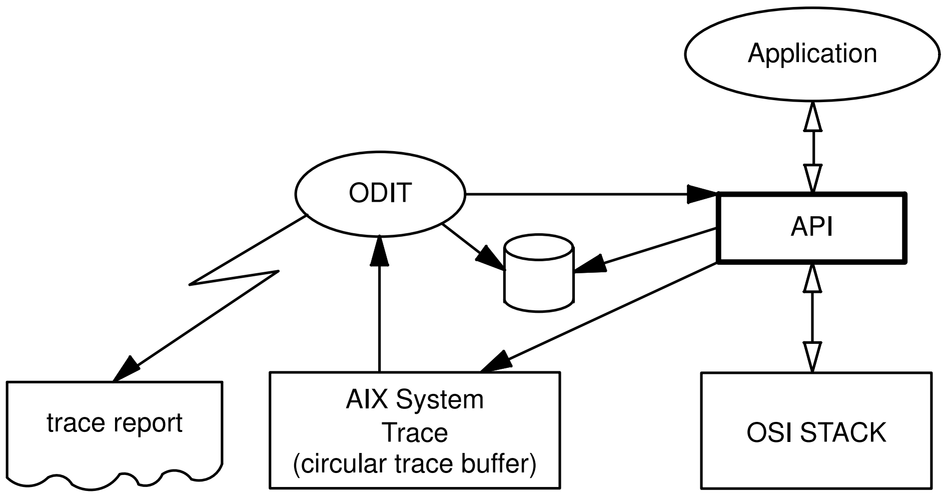


Figure 2. API trace overview

When the user selects an API trace level, the *.ositrace* file is updated in the $HOME user directory. The running API reads this file and updates its current trace level each time a new connection is opened. The same file is shared by all the APIs although each API has its own trace setting. Through the AIX system trace, a daemon captures the trace with the appropriate Hook ID and when the daemon is stopped, the trace is processed using a grammar delivered with the `osi_high` package.

Each API has a specific Hook ID. These are as follows:

| | |
|---|---|
| ROSE API | 531 |
| XAP API | 532 |
| Session API | 533 |

Traces on APIs are used by applications to identify all calls and the API interface of calls and data. Different trace levels are available to locate more precisely the interface problem. Traces are based on the AIX trace system and are accessible by all system users.

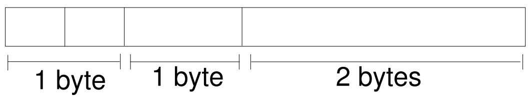
**trmad** does not enable the user to configure the trace setting of APIs. The trace display is based on the contents of the AIX circular trace buffer. It is mandatory to stop the trace recording before displaying the trace analysis.

The APIs also receive error and diagnostic codes generated by the stack layers. They either interpret this information and return it to the application using API specific format and values, or they deliver the stack code without modifying the information. See section “Error and Diagnostic Codes Analysis” on page 2-3.

A description of the codes generated by the API and of those generated by the stack is also given in the User Guide for each API (see the Bibliography at the end of this book).

## Error and Diagnostic Codes Analysis

Error and diagnostic codes are 4-bytes long. The general format for these codes is shown in the figure below.



The first byte identifies the local or remote stack component which is the originator of the error or diagnostic information. In the case of a remote entity, the diagnostic code is transmitted through PDUs. Known modules are as follows:

| | |
|---|---|
| 0x03/0x83 | local/remote X25.3 P/P |
| 0x04/0x84 | local/remote COTP |
| 0x05/0x85 | local/remote COSP |
| 0x06/0x86 | local/remote COPP |
| 0x08/0x87 | Session API |
| 0x57 | local ACSE |

The second byte is specific for each component and defines how to interpret the last two bytes. It usually defines the error context: fatal event, connection refusal, connection abort, interface error (wrong parameter, unknown address, ...)

The first two bytes are sometimes called “origsev” code (for origin/severity).

The last two bytes are a diagnostic code which is significant in a given context (i.e. the interpretation depends on the first two bytes). These two bytes are also called “causdiag” field (for cause/diagnostic).

The OSI stack packages do not provide an include file containing all the possible error codes. Nor are they listed in the documentation because they can be decoded by using the **pmaderror** command (see “**pmaderror**” on page 9-11).

The stream messages are also traced but are not decoded: retreiving an error or diagnostic code is a task for a specialist.

The codes are also accessible through the APIs (see “API Error Reporting” on page 2-2)

The codes are visible in the ICB and UCB traces where the values are clearly displayed with the same format for all the lower layer components. In the case of a fatal event, the error code is traced but not analyzed, although this can be done using ODIT or **trmad**.

### Alarm Condition

Serious problems may occur in the lower layers due to unrecoverable memory shortage, illegal memory access or abnormal stack processing. In such cases, the OSI stack generates an **alarm**. This freezes the stack lower layers which are made intentionally unusable so that the cause of the failure can be analyzed. This condition is **extremely** rare and avoids impacting the normal functioning of other applications. (It avoids, for example, a system crash).

As the lower layers are frozen, a single alarm can be traced. Several tools allow to check whether a fatal event has occurred without displaying the complete trace information.

When an alarm occurs, all the existing OSI connections are locally ended, and all local applications are notified. However, no PDUs are sent to the remote station, and the stack no longer responds to remote stations.

When using the X25.3 access method or TPAD–HPAD product, *errno* is set when accessing the driver. The *errno* values not defined in the *errno.h* include file are defined in *maduif.h*. They can be decoded using the **pmaderror command** with the **–e** option.

## Common Guidelines for Problem Analysis

If a problem occurs when using the OSI stack, it is recommended to check the following information:

1. Are the licenses available?
2. Are the drivers loaded in the kernel and what is the driver status?
3. Is there an alarm condition?
4. Are all the necessary daemons running?
5. Is there a memory shortage?
6. What configuration adapters are configured and recognized?

All this information is available using the “OSI Stack Information” option of ODIT, see “Accessing OSI Stack Information” on page 5-1. If you detect something wrong, solve the problem and then try to reproduce it. If the problem persists, a finer analysis needs to be performed.

## Connection Refusal Analysis

Apply these common guidelines to the local and (if possible) remote stations. If the problem persists, the connection may be refused because of the following causes:

1. You have an addressing problem,
2. You have a network (LAN or X25) problem,
3. You have used wrong parameters (especially for developers using APIs),
4. You have an interoperability problem,
5. You have reached the maximum number of connections supported.

If the connection is refused immediately, you have a local problem (wrong local address, wrong parameters, limit reached, interoperability problem).

If there is a time gap, the refusal may be due to a network problem or remote station problem (wrong remote address, wrong parameters, network problem).

**You must try to get the connection refusal diagnostic** (via a product message, a trace record or API parameter) and decode the code using **pmaderror**. If the answer is not clear or you cannot get this code, you may check the following points:

## Addressing problem:

1. Check whether the calling and called addresses are appropriate.
2. Check that your local and remote addresses are actually defined and located on the same network (i.e. that there is no need for a relay).
3. If NSAP addressing is used, check the addressing configuration of the local and relay systems. if a router is used, check whether it is able to route OSI CLNP protocol, and whether it is properly configured.
4. With NSAP addressing on X25, check that the NSAP is defined in the fixed part of the static rib (refer to the **osirib** command).
5. With NSAP addressing on LAN, check that the ES-IS protocol is enabled and properly configured.
6. With NSAP addressing, check that the remote system is correctly configured to reach the local system (otherwise it cannot send the response).

## Network problem:

1. Check the subnetwork status (see the *OSI Services Reference Manual*).
2. Check that the communication adapter is in available state (by using the **osiadapterinfo** command).
3. With an X25 network, check the status of the X25 layer by using the **lookx25** tool (see “lookx25” on page 9-13).
4. Check that all cables are properly connected.
5. Use the **osiping** command to validate TRA to TRA communication and to see whether another communication stack can use the same communication adapter.

## Incorrect parameter(s)

Get the proposed parameters (by checking the configuration, using the product trace, API trace or stack trace) and compare them with the possible values.

## Interoperability problem

You can use protocol analysis to understand at which level the connection is refused and which PDUs have been exchanged.When a problem occurs due to bad interoperation with another stack, the following procedure should be followed:

1. Select the OSI Protocol Analyzer menu under ODIT or else use the corresponding SMIT fastpath **odit_prot**.
2. Reset the trace buffer.
3. Run a protocol analysis.

For specific needs, the circular buffer may be too short. If this is the case, before running the protocol analysis, do the following:

1. Run the SMIT command **odit_save** corresponding to the ODIT menu “OSI Trace Save Without Analysis”.
2. Select scan mode, so that a trace file is used instead of the circular trace buffer. The default file can be selected using ESC–4.
3. Activate the application you want to test.
4. Do Ctrl C to stop trace scanning.

It is now possible to run the protocol analysis by selecting the ODIT command “Protocol Analyzer Run” or by using the corresponding SMIT fastpath **odit_pr_run**. The file input mode and the default file can then be selected using ESC–4.

For a detailed description of the menus corresponding to the various steps in this procedure, refer to “OSI Protocol Analyzer” on page 8-1.

### Limit reached

Check the configured limits for the number of connections and the transport multiplexing rate (see the *OSI Services Reference Manual*). Check also the number of connections currently used and the number of current open connections (see “Statistics Section” in “Accessing OSI Stack Information” on page 5-3).

## Connection Abort Analysis

Try to establish a new connection. If you cannot open a new connection, refer to “Connection Refusal Analysis” on page 2-4.

If you can establish new connections, but these are randomly aborted, you may have one of the following problems:

1. Local or remote memory shortage or congestion,
2. Network overload which increases the response time and leads to transport timeout,
3. X25 network problems (abnormal clear or reset).

Try to get the abort diagnostic code and interpret it using **pmaderror**. If you are not able to get this information or if the diagnostic message is not clear, you should check the following parameters:

- memory use (by using ODIT, see “Accessing OSI Stack Information” on page 5-1).
- number of transport retransmitters (by using ODIT, see “Accessing OSI Stack Information” on page 5-1).
- when operating over X25, the reset or clear packets sent or received (using **lookx25**, refer to **lookx25** on page 9-13).

If you suspect that the problem is due to an overloaded network, you may increase the transport retransmission timer and the maximum number of retransmissions (if class 4 is used).

## Trace Analysis

ODIT offers three types of traces:

- OSI stack trace,
- API traces (XAP, ROSE and Session),
- Protocol analysis.

## OSI Stack Trace

The OSI stack trace shows all the exchanges between the stack components as well as the components' internal traces.

The procedure is as follows:

1. Select the ODIT menu "OSI Layers Trace Management" or the corresponding smit fastpath **odit_st**.
2. Activate the appropriate specific traces, the trace levels. Trace also messages exchanged between layers.
3. Wait for the alarm condition to be reproduced.
4. Analyze the traces obtained in order to determine the cause of the alarm. For this step, refer to "How to Perform Trace Analysis" on page 7-52.

If you need to change a trace level, the new configuration is immediately taken into account. If you use scan mode, new information is traced according to the new configuration. For further information, refer to "OSI Stack Layers Trace Management" on page 7-1 and for trace samples to "Trace Report" on page A-1.

## API Traces

XAP, ROSE and Session API traces show the exchanges between the application and the OSI stack and also the API internal processing.

These traces can be used by application developers to diagnose problems in the use of APIs as well as problems with Bull applications built on top of these APIs. The trace grammar required to decode API traces is not automatically installed. Refer to "ODIT Installation" on page 3-1.

The following procedure must be followed:

1. Select the OSI API Trace Management menu under ODIT or else use the corresponding SMIT fastpath **odit_api**.
2. Activate the trace for the API by setting the appropriate trace levels.
3. Start the trace.
4. Activate the application for which the interfacing problem occurred.
5. Stop the trace.
6. Analyze the trace.

For a detailed description of the menus corresponding to the various steps in this procedure and an example of the trace produced, refer to "API Trace Management" on page 6-1.

Unlike the OSI stack trace, the new trace settings are only taken into account for new connections. Furthermore, it is mandatory to start and stop the trace recording before displaying the results. It is not possible to have a continuous display.

## Protocol Analysis

The protocol analysis decodes the PDUs exchanged with remote stations. Many filters are available to focus only on the traces relevant to the analysis of a specific problem.

## Alarm Condition

The commands **osiinfo** or **errpt** will provide an explanation for an OSI stack alarm. The most frequent causes are lack of memory or incorrect configuration of X25 boards.

Use ODIT or the **trmad** command to detect lack of memory: See “Accessing OSI Stack Information” on page 5-1 and **trmad** (Memory Analysis) on page 9-10.

Use **lookx25** to diagnose X25 problems. See “**lookx25**” on page 9-13.

You will need to perform **osiunload** followed by **osiload** before you can use the stack again.

In all cases, an alarm is a malfunction of the OSI stack and should be reported to Technical Support who will provide assistance.

## Crash Condition

In the unlikely event of a system crash, the following procedure should be followed:

1. Try to reproduce the problem. If this is feasable, proceed with the next steps.
2. Perform a memory dump after the crash has occurred.
3. Run the **osisnap** command to gather all system and stack information. See “**osisnap**” on page 9-23.
4. Contact Technical Support for assistance.

# Chapter 3. ODIT Installation

This Chapter describes the contents of the ODIT package and how to install it with the System Configurator.

## ODIT Packaging

The commands and SMIT menus described in this manual are packaged in several packages and filesets.

### OSI Stack Framework

The `osi_frame.odit` fileset contains the SMIT menus and the **osiinfo**, **ositrace**, **osiprot**, **ositrcstart**, **ositrcstop** and **ositrcrpt** commands.

The `osi_frame.rte` fileset contains the **trmad** and **pmaderror** commands.

### OSI Stack Lower Layers

The `osi_low.rte` fileset contains the **lookx25** command.

### OSI SNAP

The `osisnap` package contains the **osisnap** command.

## Installation Procedure

ODIT is automatically installed at the same time as the OSI Stack packages, using SMIT. Refer to "OSI Stack Installation" in *OSI Services Reference Manual*.

## License Control

All the tools covered by this manual are license free.

# Chapter 4. Accessing ODIT

## Introducing the ODIT Menu Structure

The OSI Diagnostic Interactive Toolkit is reached through the OSI Networking menu of the OSI Stack configurator.

The ODIT menus are structured as follows, the indentations indicating the level in the structure.

### OSI Diagnostic Interactive Toolkit on page 4-3

#### OSI Stack Information on page 4-4

#### OSI API Trace Management on page 4-5

- XAP API Trace Management on page 4-6:
  - Change/Show XAP API Trace Levels
  - Start/Stop Trace
  - Trace Analysis
- ROSE API Trace Management on page 4-6:
  - Change/Show ROSE API Trace Levels
  - Start/Stop Trace
  - Trace Analysis
- Session API Trace Management on page 4-7:
  - Change/Show Session API Trace Levels
  - Start/Stop Trace
  - Trace Analysis

#### OSI Layers Trace Management on page 4-8

- List Trace Configuration on page 4-8
- Reset Trace Configuration on page 4-9
- General Trace Configuration on page 4-9
- Change/Show OSI Layers Trace Levels on page 4-10:
  - Change/Show ACSE Trace Levels
  - Change/Show COPP Trace Levels
  - Change/Show COSP Trace Levels
  - Change/Show COTP Access Method Trace Levels
  - Change/Show COTP Driver Trace Levels
  - Change/Show COTP Trace Levels
  - Change/Show X25.3 Access Method Trace Levels
  - Change/Show X25.3 Trace Levels
  - Change/Show X25.2 Trace Levels
  - Change/Show CLNP Trace Levels
  - Change/Show ES–IS Trace Levels
  - Change/Show LLC Trace Levels
  - Change/Show TIMER Trace Levels
  - Change/Show Administration Trace Levels
  - Change/Show MAD Trace Levels
- Reset Trace Buffer on page 4-11
- Trace Analysis on page 4-11

## ODIT Main Menu

The ODIT main menu gives access to five submenus:

- OSI Stack Information, see page 5-1,
- OSI API Trace Management, see page 6-1,
- OSI Layers Trace Management, see page 7-1,
- OSI Protocol Analyzer, see page 8-1,
- OSI Trace Save Without Analysis, see page 7-54,

The ODIT main menu, submenus and menu functions are presented hereafter.

# ODIT – Quick Guide

This guide presents the ODIT menus in graphic form, showing interdependencies, FastPath access, paragraph and page references.

The functions are sorted in alphabetical order in the general index under the entry “Menu functions”.

The **ODIT main menu** displayed when running ’**smit odit**’ is illustrated in the following figure.

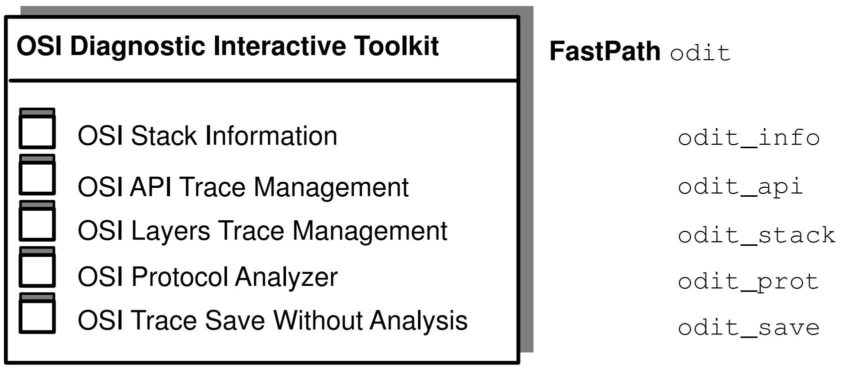


Figure 3.  ODIT Main Menu

**Fastpath**: `odit`

For a description of the ODIT Main Menu, refer to Starting ODIT on page 4-17.

The five submenus of the ODIT Main Menu are described in the following pages.

**The “OSI Stack Information” submenu item is illustrated in the following figure.**

**Fastpath**: `odit_info`

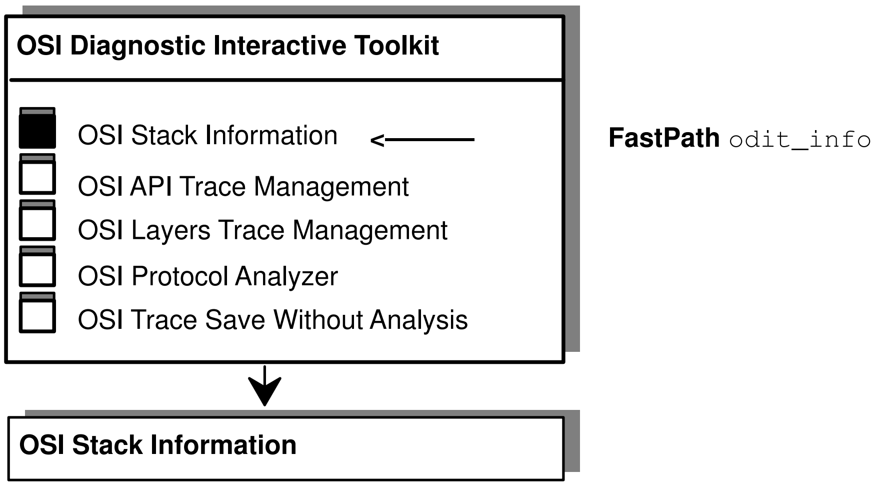


Figure 4. OSI Stack Information Menu Function

For an example of the information displayed on selecting the OSI Stack Information menu option, refer to “Accessing OSI Stack Information” on page 5-1.

**The “OSI API Trace Management” submenu is illustrated in the following figure.**

**Fastpath**: `odit_api`

The content of this menu varies depending on the packages present: osi_high package (ROSE and Session) alone or with the xap_api package. The xap_api package cannot be installed alone because the osi_high package is an installation prerequisite. The availability of the submenus described below is also dependent on whether or not the associated packages are present.

For a detailed description of the functions offered by this submenu, refer to Accessing OSI API Trace Management on page 6-1.

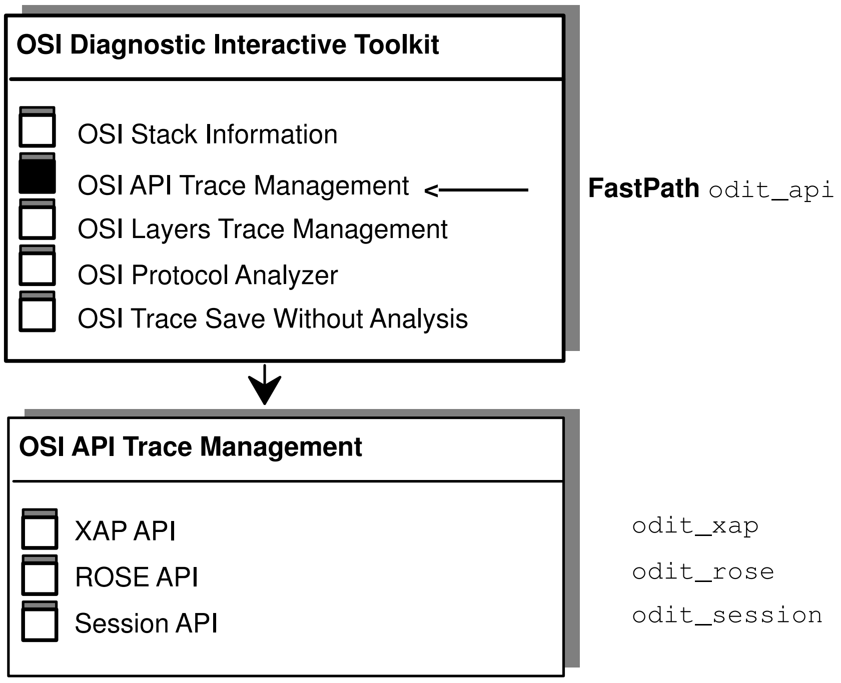


Figure 5.  OSI API Trace Management Submenu

**The "XAP API Trace Management" submenu is illustrated in the following figure.**

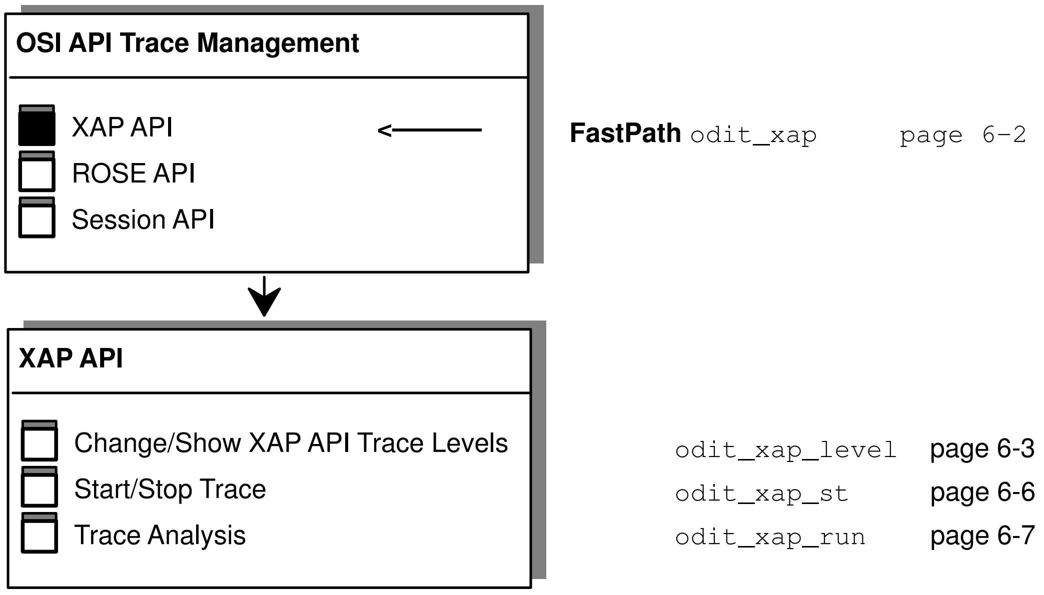


Figure 6. XAP API Trace Management Submenu

**The "ROSE API" submenu is illustrated in the following figure.**

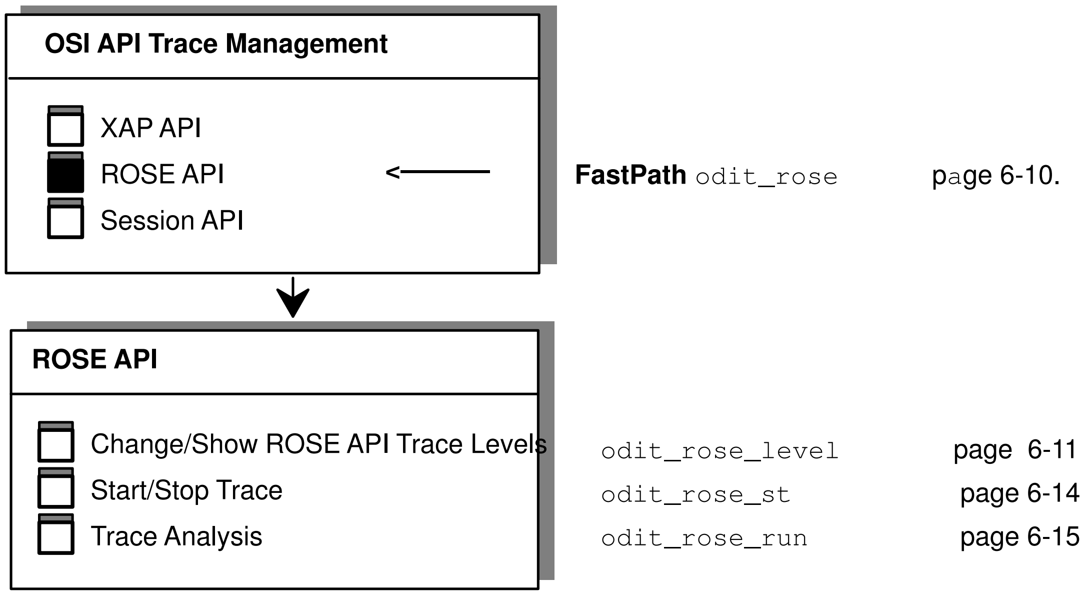


Figure 7. ROSE API Trace Management Submenu

**The “Session API” submenu is illustrated in the following figure.**

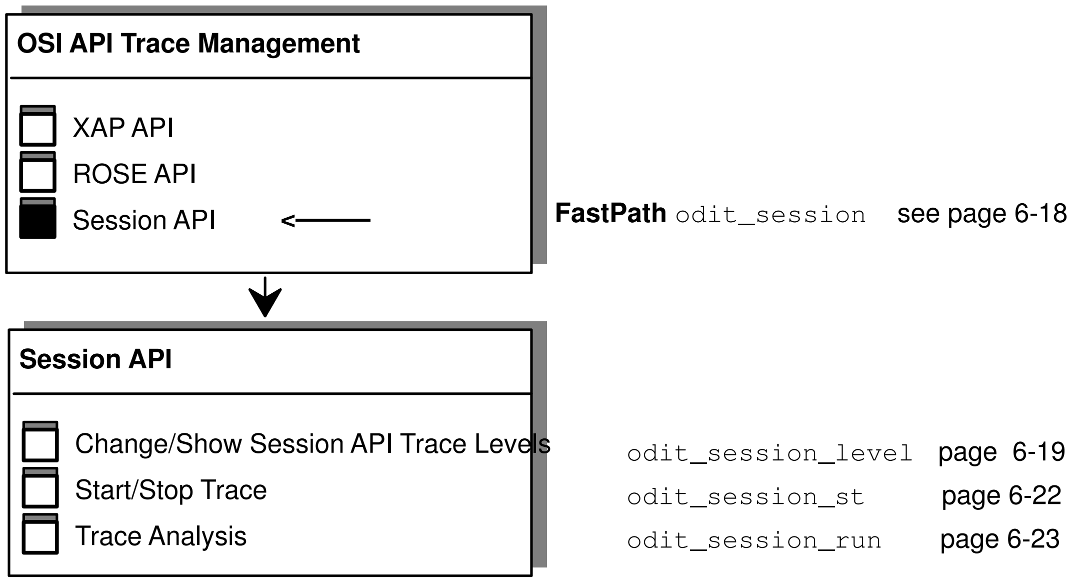

OSI API Trace Management
XAP API
ROSE API
Session API
FastPath odit_session see page 6-18
Session API
Change/Show Session API Trace Levels
Start/Stop Trace
Trace Analysis
odit_session_level page 6-19
odit_session_st page 6-22
odit_session_run page 6-23

Figure 8. Session API Trace Management Submenu

**The “OSI Layers Trace Management” submenu is illustrated in the following figure.**
**Fastpath**: `odit_stack`

For a detailed description of the functions offered by this submenu, refer to Accessing OSI Stack Internal Trace Management on page 7-1.

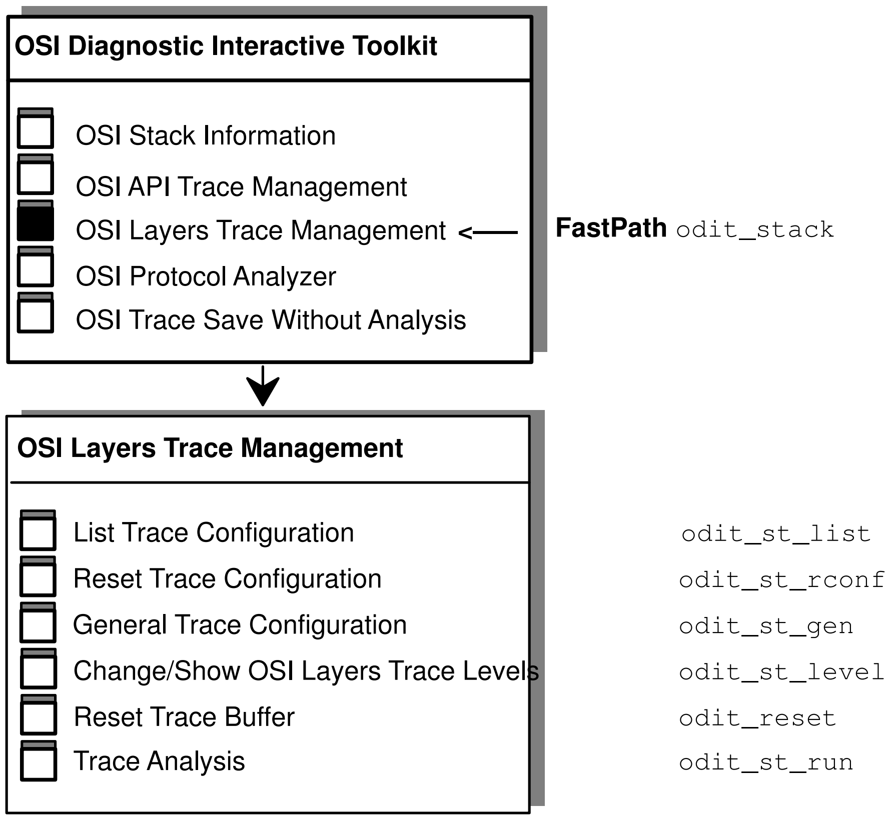


Figure 9. OSI Layers Trace Management Submenu

**The “List Trace Configuration” menu item is illustrated in the following figure.**
**Fastpath**: `odit_st_list`

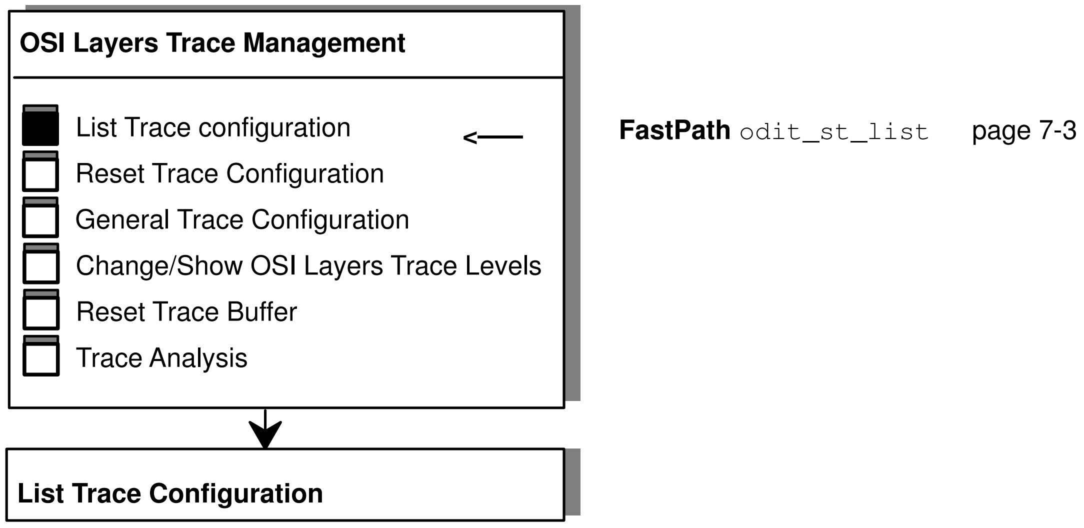

OSI Layers Trace Management
List Trace configuration
Reset Trace Configuration
General Trace Configuration
Change/Show OSI Layers Trace Levels
Reset Trace Buffer
Trace Analysis
FastPath odit_st_list
page 7-3
List Trace Configuration

Figure 10. List Trace Configuration Menu Function

**The “Reset Trace Configuration” menu item is illustrated in the following figure.**

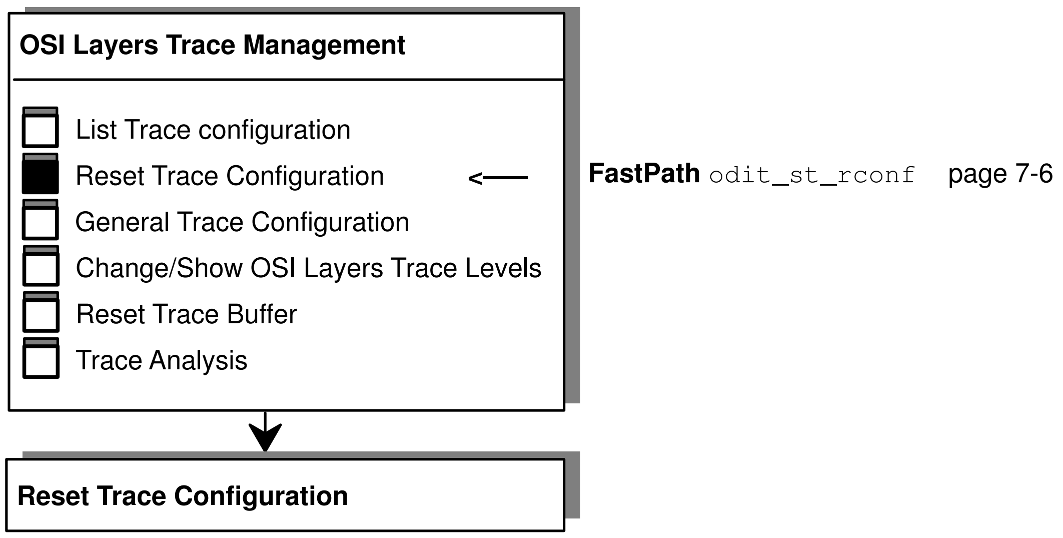

OSI Layers Trace Management
List Trace configuration
Reset Trace Configuration
General Trace Configuration
Change/Show OSI Layers Trace Levels
Reset Trace Buffer
Trace Analysis
<—
FastPath odit_st_rconf page 7-6
Reset Trace Configuration

Figure 11. Reset Trace Configuration Menu Function

**The “General Trace Configuration” submenu is illustrated in the following figure.**

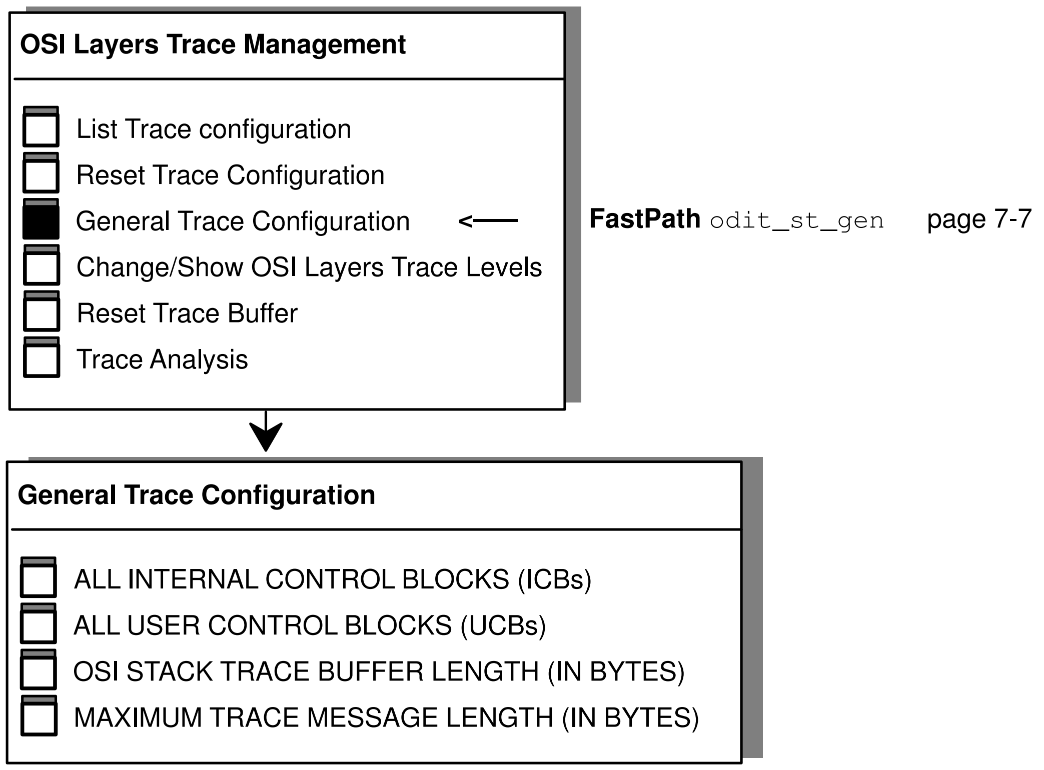

OSI Layers Trace Management
List Trace configuration
Reset Trace Configuration
General Trace Configuration
Change/Show OSI Layers Trace Levels
Reset Trace Buffer
Trace Analysis
<—
FastPath odit_st_gen page 7-7
General Trace Configuration
ALL INTERNAL CONTROL BLOCKS (ICBs)
ALL USER CONTROL BLOCKS (UCBs)
OSI STACK TRACE BUFFER LENGTH (IN BYTES)
MAXIMUM TRACE MESSAGE LENGTH (IN BYTES)

Figure 12. OSI Layers General Trace Configuration Submenu

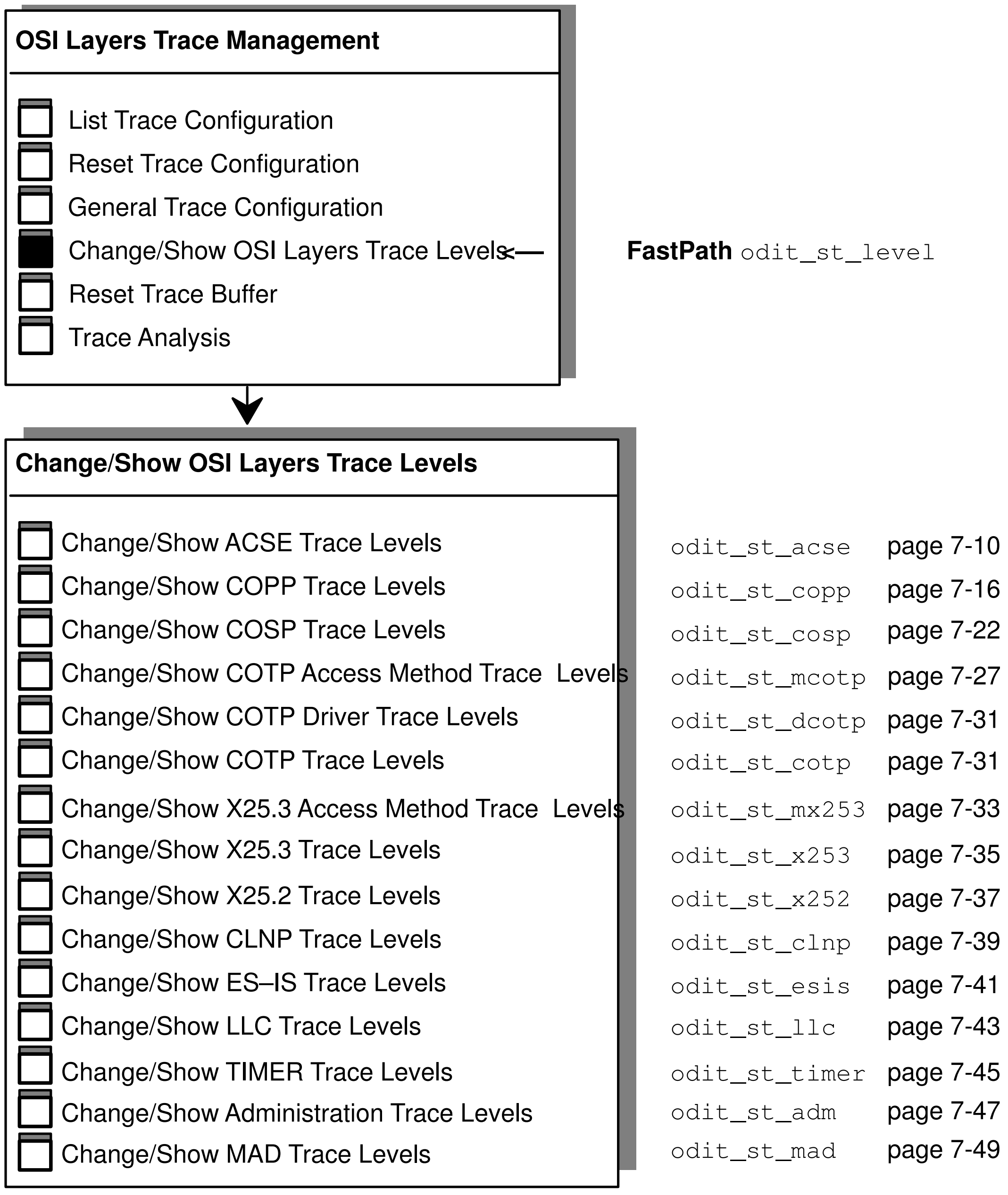
**The “Change/Show OSI Layers Trace Levels” submenu is illustrated in the following figure.**

**Fastpath**: `odit_st_level`

The first three functions in this submenu only appear when `osi_high` is installed.

The following nine functions (from COTP Access Method Trace Levels to LLC Trace Levels) only appear when `osi_low` is installed.

For a detailed description of the functions offered by this submenu, refer to How to Change or Show OSI Layers Trace Levels on page 7-9.



OSI Layers Trace Management
List Trace Configuration
Reset Trace Configuration
General Trace Configuration
Change/Show OSI Layers Trace Levels
Reset Trace Buffer
Trace Analysis
FastPath odit_st_level
Change/Show OSI Layers Trace Levels
Change/Show ACSE Trace Levels odit_st_acse page 7-10
Change/Show COPP Trace Levels odit_st_copp page 7-16
Change/Show COSP Trace Levels odit_st_cosp page 7-22
Change/Show COTP Access Method Trace Levels odit_st_mcotp page 7-27
Change/Show COTP Driver Trace Levels odit_st_dcotp page 7-31
Change/Show COTP Trace Levels odit_st_cotp page 7-31
Change/Show X25.3 Access Method Trace Levels odit_st_mx253 page 7-33
Change/Show X25.3 Trace Levels odit_st_x253 page 7-35
Change/Show X25.2 Trace Levels odit_st_x252 page 7-37
Change/Show CLNP Trace Levels odit_st_clnp page 7-39
Change/Show ES–IS Trace Levels odit_st_esis page 7-41
Change/Show LLC Trace Levels odit_st_llc page 7-43
Change/Show TIMER Trace Levels odit_st_timer page 7-45
Change/Show Administration Trace Levels odit_st_adm page 7-47
Change/Show MAD Trace Levels odit_st_mad page 7-49

Figure 13. Change/Show OSI Layers Trace Levels Submenu

**The “Reset Trace Buffer” menu item is illustrated in the following figure.**

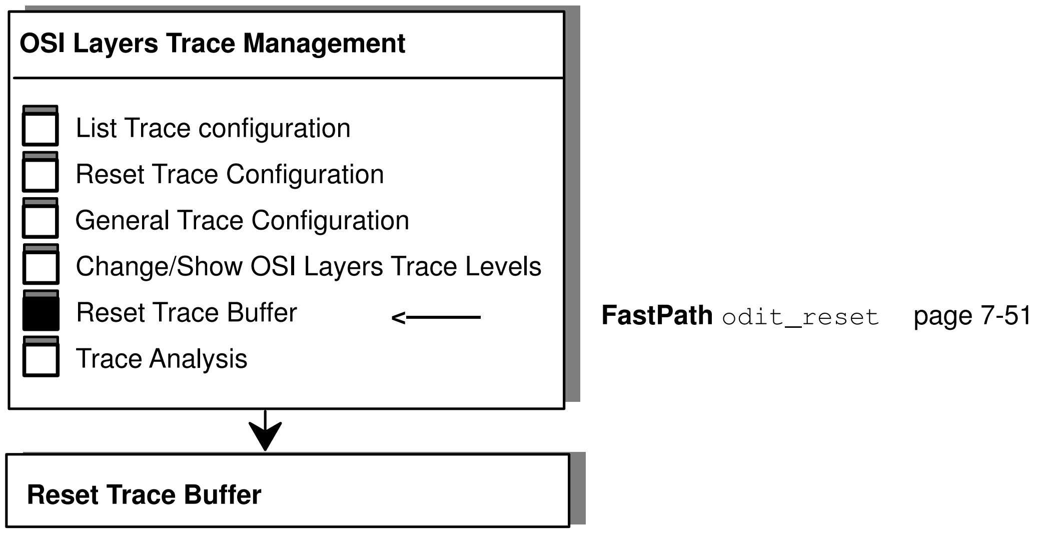

OSI Layers Trace Management
List Trace configuration
Reset Trace Configuration
General Trace Configuration
Change/Show OSI Layers Trace Levels
Reset Trace Buffer
Trace Analysis
FastPath odit_reset page 7-51
Reset Trace Buffer

Figure 14. OSI Layers Reset Trace Buffer Menu Function

**The “Trace Analysis” submenu is illustrated in the following figure.**

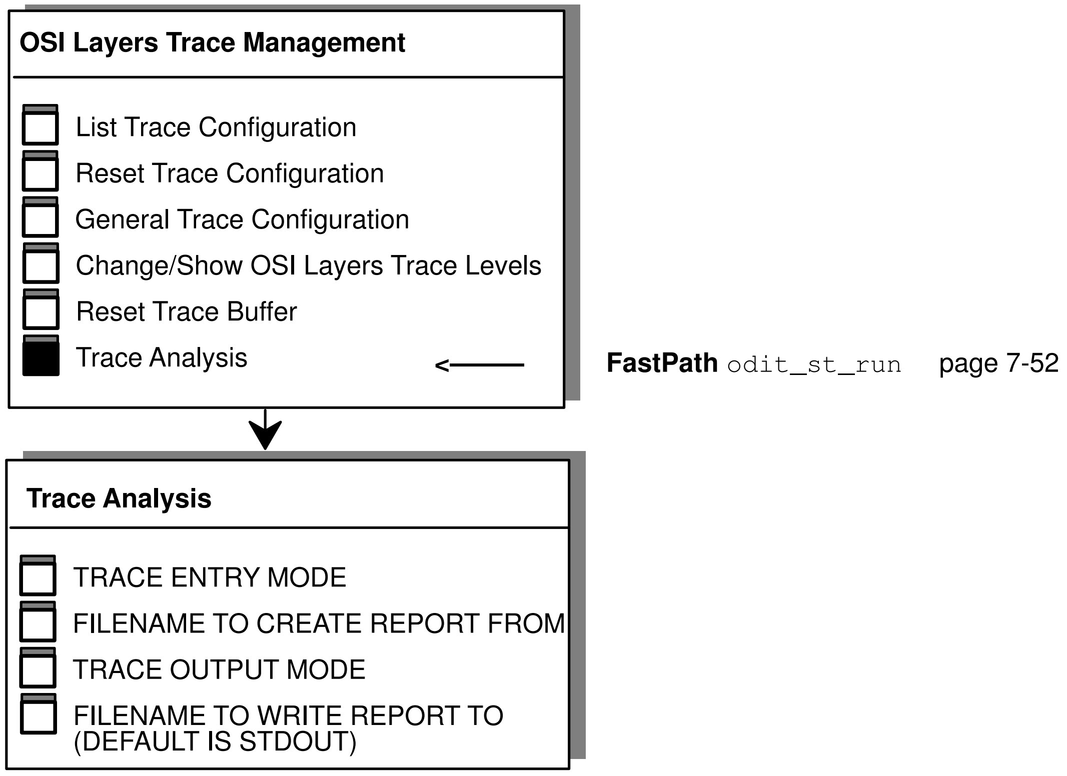

OSI Layers Trace Management
List Trace Configuration
Reset Trace Configuration
General Trace Configuration
Change/Show OSI Layers Trace Levels
Reset Trace Buffer
Trace Analysis
FastPath odit_st_run page 7-52
Trace Analysis
TRACE ENTRY MODE
FILENAME TO CREATE REPORT FROM
TRACE OUTPUT MODE
FILENAME TO WRITE REPORT TO (DEFAULT IS STDOUT)

Figure 15. OSI Layers Trace Analysis Submenu

**The “OSI Protocol Analyzer” submenu is illustrated in the following figure.**

**Fastpath**: `odit_prot`

For a detailed description of the functions offered by this submenu, refer to Accessing the OSI Protocol Analyzer on page 8-1.

The access for the submenus of the OSI Protocol Analyzer submenu is provided in the following pages.

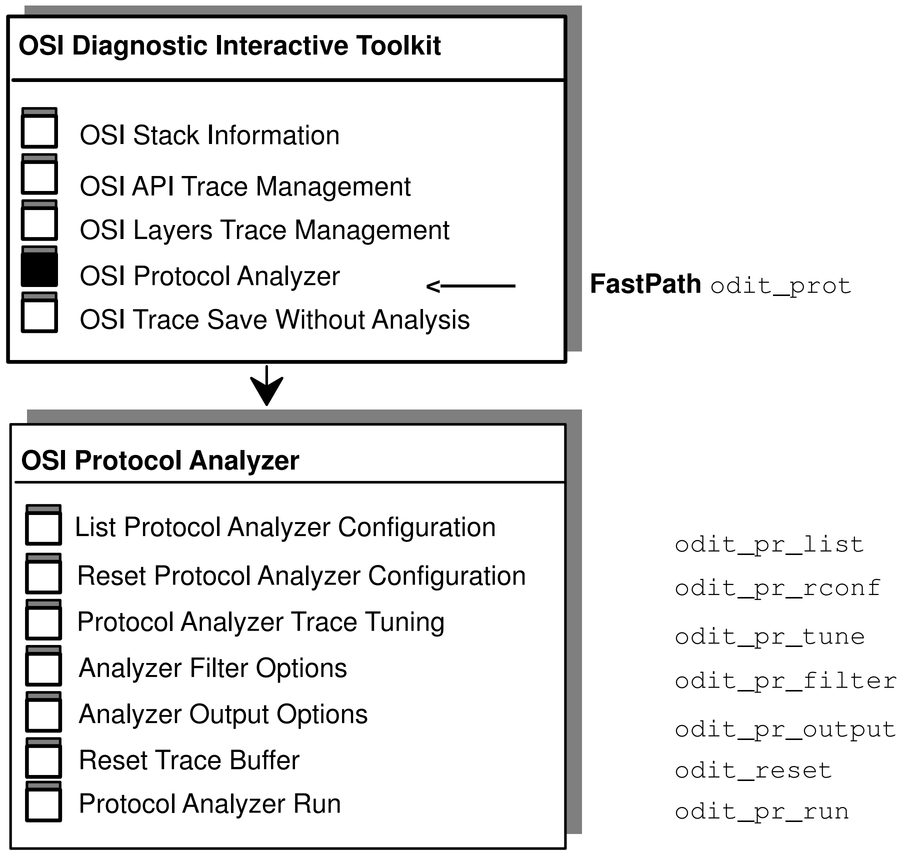


Figure 16. OSI Protocol Analyzer Submenu

**The “List Protocol Analyzer Configuration” menu item is illustrated in the following figure.**

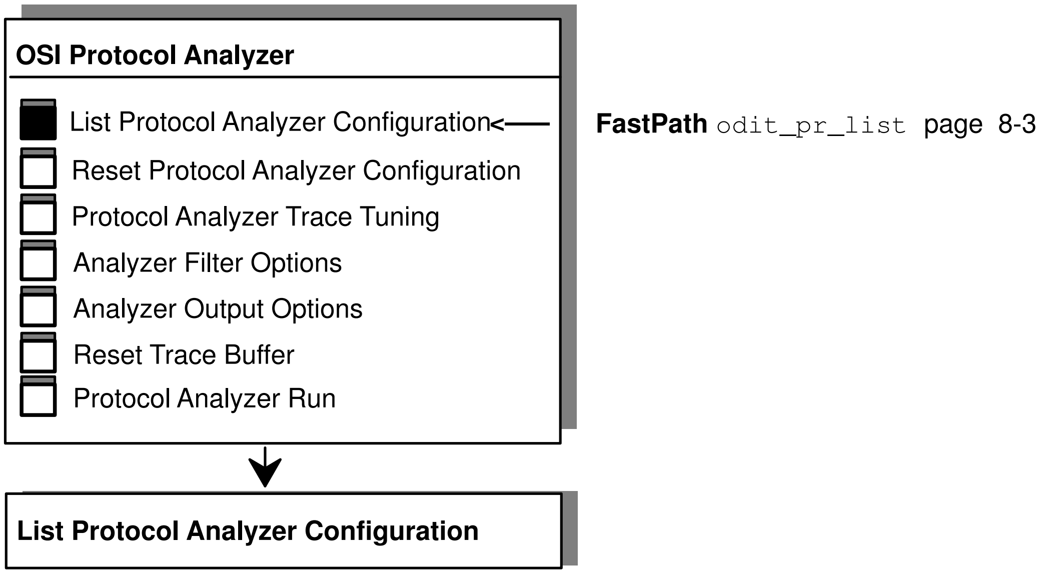

OSI Protocol Analyzer
List Protocol Analyzer Configuration
Reset Protocol Analyzer Configuration
Protocol Analyzer Trace Tuning
Analyzer Filter Options
Analyzer Output Options
Reset Trace Buffer
Protocol Analyzer Run
FastPath odit_pr_list page 8-3
List Protocol Analyzer Configuration

Figure 17. List Protocol Analyzer Configuration Menu Function

**The “Reset Protocol Analyzer Configuration” menu item is illustrated in the following figure.**

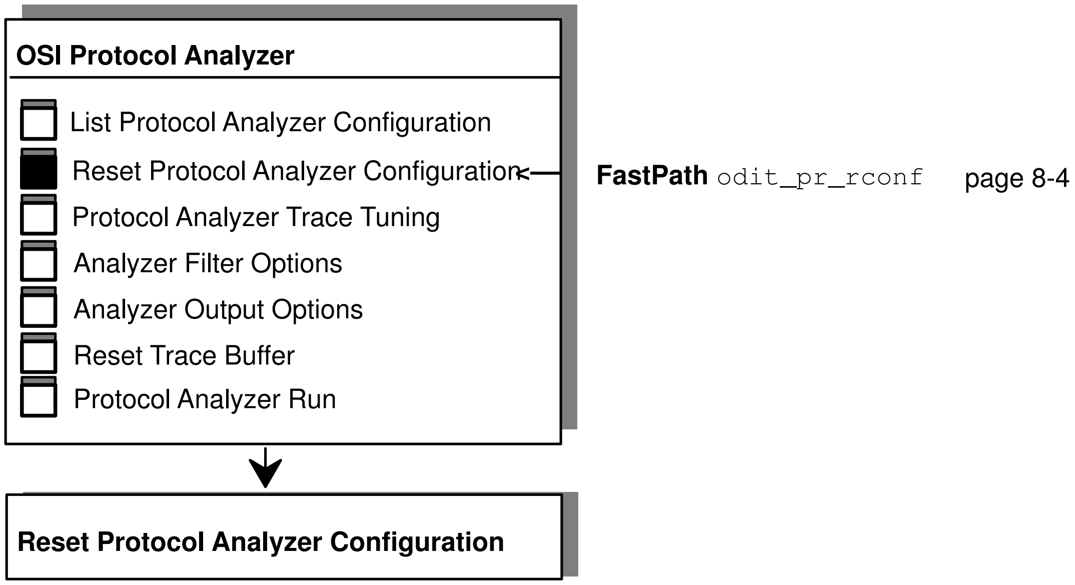

OSI Protocol Analyzer
List Protocol Analyzer Configuration
Reset Protocol Analyzer Configuration
Protocol Analyzer Trace Tuning
Analyzer Filter Options
Analyzer Output Options
Reset Trace Buffer
Protocol Analyzer Run
FastPath odit_pr_rconf page 8-4
Reset Protocol Analyzer Configuration

Figure 18. Reset Protocol Analyzer Configuration Menu Function

**The “Protocol Analyzer Trace Tuning” submenu is illustrated in the following figure.**

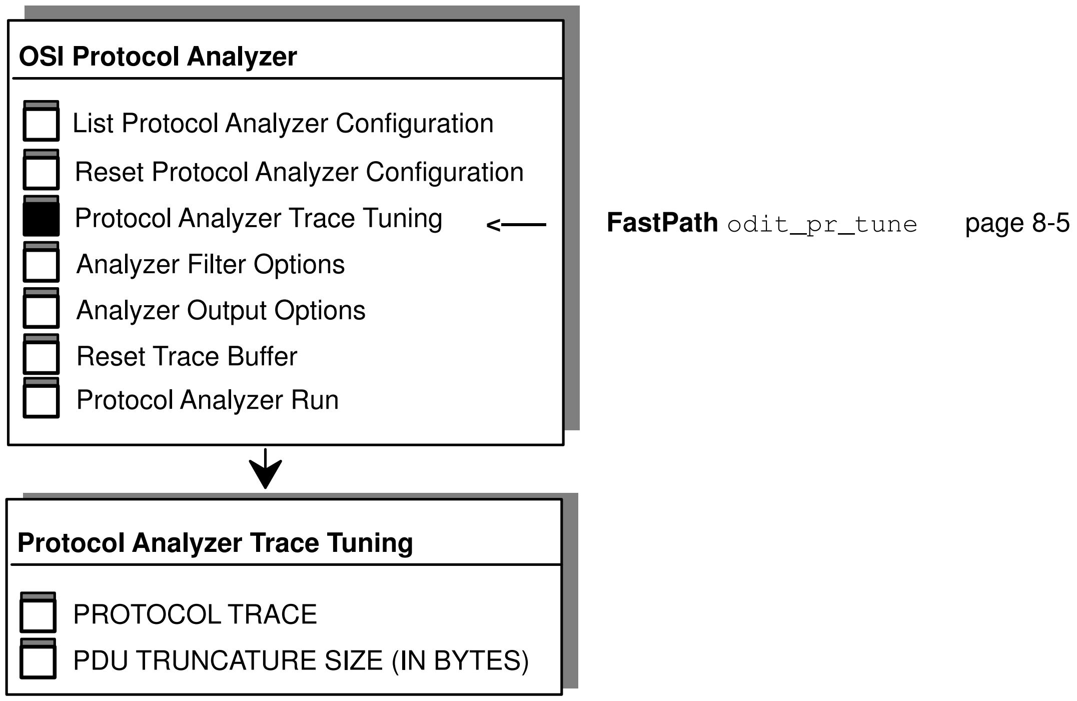

OSI Protocol Analyzer
List Protocol Analyzer Configuration
Reset Protocol Analyzer Configuration
Protocol Analyzer Trace Tuning
Analyzer Filter Options
Analyzer Output Options
Reset Trace Buffer
Protocol Analyzer Run
FastPath odit_pr_tune page 8-5
Protocol Analyzer Trace Tuning
PROTOCOL TRACE
PDU TRUNCATURE SIZE (IN BYTES)

Figure 19. OSI Protocol Analyzer Trace Tuning Submenu

**The “Analyzer Filter Options” submenu is illustrated in the following figure.**

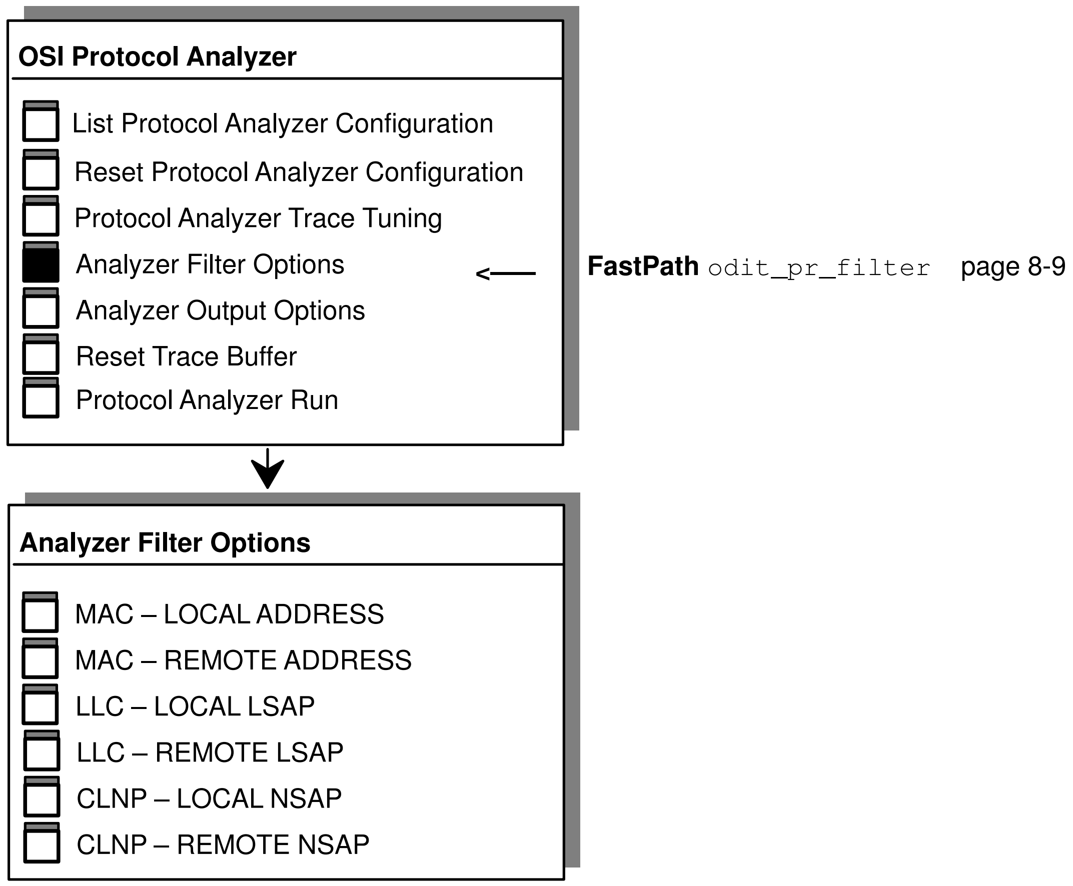


Figure 20. OSI Protocol Analyzer Filter Options Submenu

**The “Analyzer Output Options” submenu is illustrated in the following figure.**

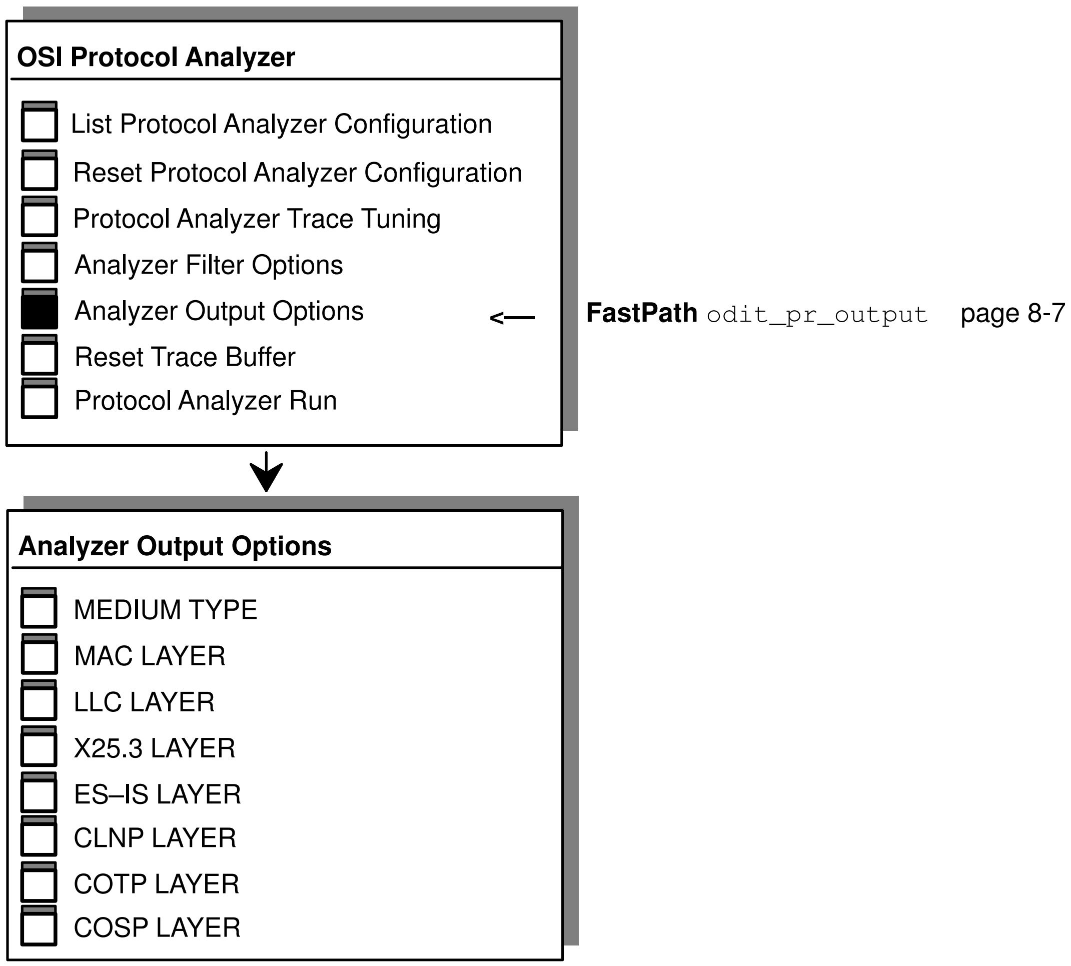


Figure 21. OSI Protocol Analyzer Output Options Submenu

**The “Reset Trace Buffer” menu item is illustrated in the following figure.**

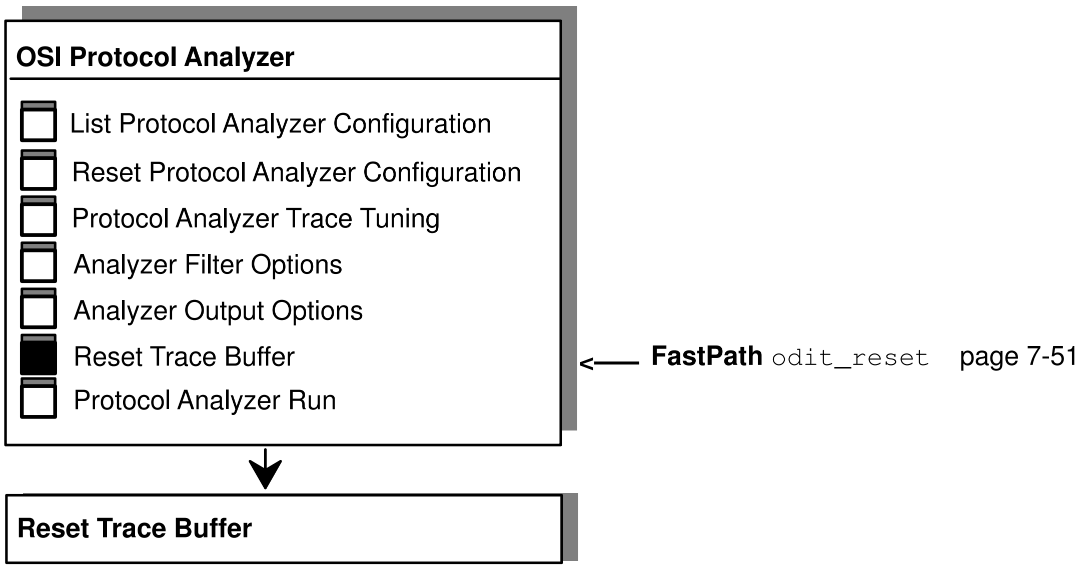

OSI Protocol Analyzer
List Protocol Analyzer Configuration
Reset Protocol Analyzer Configuration
Protocol Analyzer Trace Tuning
Analyzer Filter Options
Analyzer Output Options
Reset Trace Buffer
Protocol Analyzer Run
FastPath odit_reset page 7-51
Reset Trace Buffer

Figure 22. Reset Trace Buffer Menu Function

**The “Protocol Analyzer Run” submenu is illustrated in the following figure.**

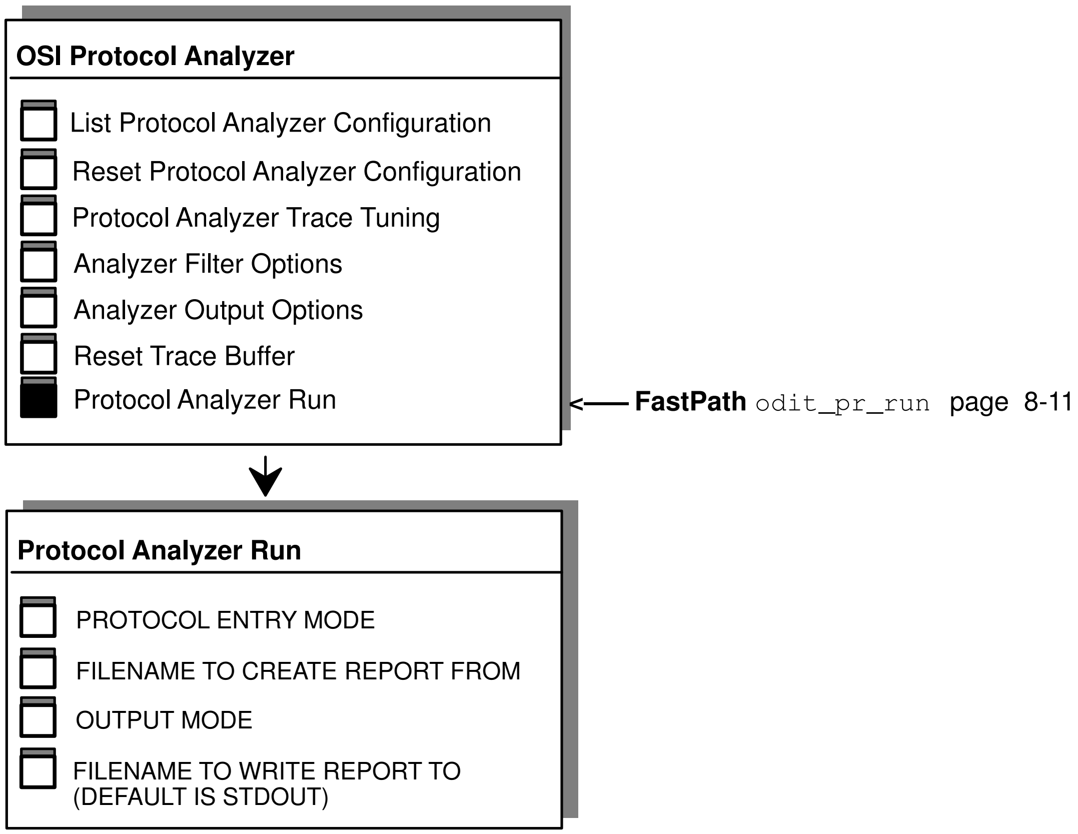

OSI Protocol Analyzer
List Protocol Analyzer Configuration
Reset Protocol Analyzer Configuration
Protocol Analyzer Trace Tuning
Analyzer Filter Options
Analyzer Output Options
Reset Trace Buffer
Protocol Analyzer Run
FastPath odit_pr_run page 8-11
Protocol Analyzer Run
PROTOCOL ENTRY MODE
FILENAME TO CREATE REPORT FROM
OUTPUT MODE
FILENAME TO WRITE REPORT TO (DEFAULT IS STDOUT)

Figure 23. OSI Protocol Analyzer Run Submenu

**The “OSI Trace Save Without Analysis” submenu is illustrated in the following figure.**

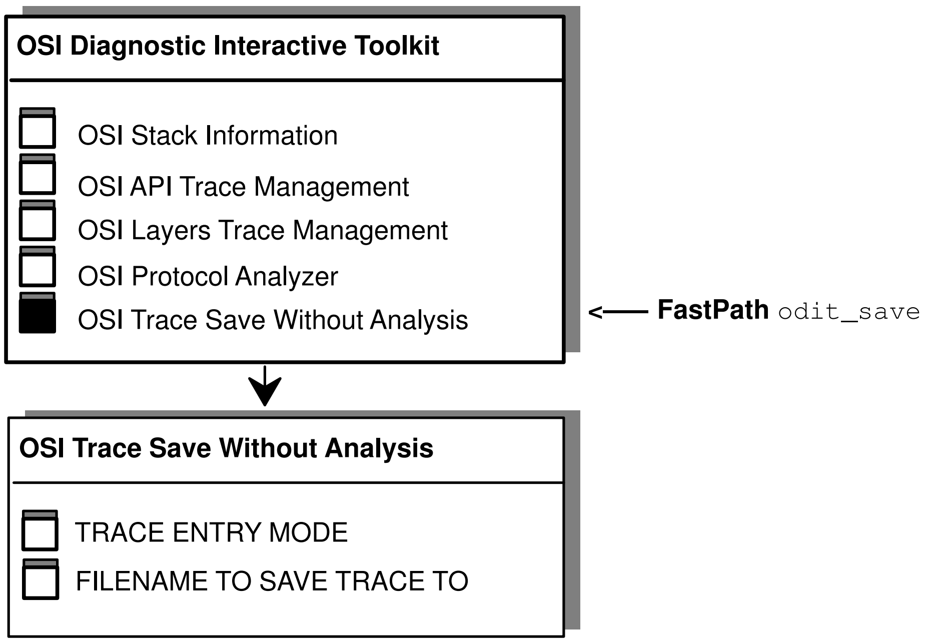

OSI Diagnostic Interactive Toolkit
OSI Stack Information
OSI API Trace Management
OSI Layers Trace Management
OSI Protocol Analyzer
OSI Trace Save Without Analysis
FastPath odit_save
OSI Trace Save Without Analysis
TRACE ENTRY MODE
FILENAME TO SAVE TRACE TO

Figure 24. OSI Trace Save Without Analysis Submenu

## Starting ODIT

### Access

From the OSI Networking menu, select:

```
OSI Diagnostic Interactive Toolkit
```

**FastPath**: `odit`

### Overview

This menu gives access to the OSI Stack trace management tool.

```
OSI Diagnostic Interactive Toolkit

OSI Stack Information
OSI API Trace Management
OSI Layers Trace Management
OSI Protocol Analyzer
OSI Trace Save Without Analysis
```

### Description

OSI Stack Information
: To display the major OSI Stack statistics and administrative counter values. See Chapter 5 “OSI Stack Information”.

OSI API Trace Management
: To display current API trace levels and reset these trace levels. See Chapter 6 “API Trace Management”.

OSI Layers Trace Management
: To display and configure OSI layers trace levels, to reset the trace buffer and to analyze the trace. See Chapter 7 “OSI Stack Layers Trace Management”.

OSI Protocol Analyzer
: To set up and run the OSI Protocol Analyzer. See Chapter 8 “OSI Protocol Analyzer”.

OSI Trace Save Without Analysis
: To save the OSI Stack trace to a file in binary format. See “How to Save a Trace to a File” on page 7-54.

# Chapter 5. OSI Stack Information

This chapter describes the ODIT OSI Stack information facility. This facility provides:

- a description of the packs installed,
- a list of running processes,
- statistics on OSI layers,
- a list of declared and configured subnets.

## Accessing OSI Stack Information

### Access

From the OSI Diagnostic Interactive Toolkit menu, select:

```
ODIT - OSI Stack information
```

**FastPath**: `odit_info`

**Command**: `osiinfo`

### Description

Two examples of the information displayed are provided below. The first (see Figure 1) illustrates the usual display whereas the second (see Figure 2) illustrates an alarm condition caused by a problem on the communication stack.

```
=========================LICENSE & DRIVER STATUS=========================
   osi_high       Unlicensed    Unloaded
   max3           Licensed
   osi_low        Licensed      Loaded
   osi_frame                    Loaded
==============================DAEMONS====================================
   osilinkd         Running
   osinetlsd        Running
   osiribd          Running
   dat_x25          Running
==============================STATISTICS=================================
OSI Driver: nbOpen ......       2   Memory Use ...   14% (Max= 81920 Kb)

COSP Layer: nbOpen .....    1/2049  NbCnx ........  0/1024
            nbPDUSend ..      767   NbPDURece ....     534
            nbOctetSend     568804  nbOctetRece ..   20170
            nbCnxRefused       10   nbCnxAborted .      56

TPI Layer:  nbOpen .....    504/2048

COTP Layer: nbCnx .....     574/1024
            nbPDUSend ..     1750   nbPDURece ....   1619
            nbOctetSend     577382  nbOctetRece ..  26504
            nbRetry ....       13   nbZeroCredit .      0
================================BOARDS===================================
   IDENT    TYPE    SUBDOMAIN   ADDRESS         LOCATION      STATE
     1      ETH       10        02608c2c8b79       --          ON
```

Figure 1. Typical ODIT information display

```
================================ALARM====================================
        Module          : Communicator
        Func/Cause      :
        Error/Diag      : Error in freeing a chain
========================LICENSE & DRIVER STATUS==========================
   osi_high       Licensed  Loaded
   max3           Licensed
   osi_low        Licensed  Alarm
   osi_frame      Licensed  Alarm
==============================DAEMONS====================================
   osilinkd             Running
   osinetlsd            Running
   osiribd              Running
   dat_x25              Running
==============================STATISTICS=================================
OSI Driver: nbOpen ......       2   Memory Use ...   0% (Max= 81920 Kb)

COSP Layer: nbOpen ......   1/2049  nbCnx ........  0/1024
            nbPDUSend ...     767   nbPDURece ....     534
            nbOctetSend .   568804  nbOctetRece ..   20170
            nbCnxRefused       10   nbCnxAborted .      56

TPI Layer:  nbOpen ......   1/2048

COTP Layer: nbCnx ......    0/1024
            nbPDUSend ...    1750   nbPDURece ....   1619
            nbOctetSend .   577382  nbOctetRece ..  26504
            nbRetry ....       13   nbZeroCredit .      0
```

Figure 2. ODIT information display with an alarm condition

The various sections of these displays are described in detail below.

## Alarm Section

The first section of the screen is displayed only if an alarm condition occurs. In such a case, there is a message to indicate the probable cause of the alarm.

## License and Driver Status Section

This section indicates the license status for each installed Licensed Program Product (LPP).

**Note:** The license status displayed by **osiinfo** is the reflection of the machine's licensing status and the current running licenses on the OSI stack may be different because the stack obtains this licensing data from a daemon at five minute intervals. Thus, if you install a new product, the stack will need up to five minutes in order to take this new product license into account.

The format of the display is as follows:

```
<component> <license status> <driver status>
```

| | |
|---|---|
| component | `osi_high` for the OSI upper layers<br>`max 3` for the X25.3 access method<br>`osi_low` for the OSI lower layers<br>`osi_frame` for the OSI framework. |
| license status | `Licensed` means that license is available<br>`Unlicensed` means that license is not available<br>`<blank>` means license free. |
| driver status | `Loaded` means available for use<br>`Unloaded` means not available for use<br>`Alarm` means a fatal error has occurred (see Alarm section of the display). |

If a component is in Alarm state, the OSI stack must be de-activated and re-activated (using the **osiunload** and **osiload** commands respectively) or the system must be rebooted.

## Daemons Section

This section displays the status of the daemons used by the OSI stack. The status is either "Running" or "Not Running". The daemons and their normal status are listed below.

| | |
|---|---|
| **osilinkd** | Used to maintain links between streams drivers. Must be running if `osi_high` is loaded. If this is not the case, the OSI stack needs to be de-activated and then re-activated (using **osiunload** and **osiload** respectively) or else the system must be rebooted. |
| **osinetlsd** | Checks the stack licenses. Must be running. If not so, restart the daemon by using the **osinetlsd** command. |
| **osiribd** | Manages the static RIB. If `osi_low` is loaded and static routing is used, the daemon should be running. If not so, restart it using the **osiribd** command. |
| **dat_x25** | Manages streams between the OSI stack and the HiSpeed WAN communications adapter. Must be running if a HiSpeed WAN communications adapter is used by the OSI stack. |

## Statistics Section

The "Memory Use" field indicates the maximum memory used compared with the maximum allowed in the current configuration. If your are close to 100%, it is recommended to increase the maximum memory allowed (see "OSI Stack Configuration" in the *OSI Services Reference Manual*).

The "nbOpen" field indicates the current number of open connections on the specified driver.

The "nbPDUSend" field indicates the current number of PDUs sent.

The "nbPDUReceived" field indicates the current number of PDUs received.

The "nbOctetSend" field indicates the current number of octets sent.

The "nbOctetReceived" field indicates the current number of octets received.

The "nbCnxRefused" field indicates the current number of COSP connections refused.

The "nbCnxAborted" field indicates the current number of COSP connections aborted.

The "nbRetry" field indicates the current number of COTP retransmissions. This parameter is incremented each time a PDU is retransmitted. Thus, this field provides a quality measure for the network. If the number is high, it indicates poor network reliability or bad calibration of the transport retransmission timer.

The "nbZeroCredit" field indicates the current number of COTP zero credit sent. This parameter is incremented each time a COTP layer needs more time to handle PDUs. Thus, this field provides a measure of the machine's velocity. If the number is high, it indicates a machine overbooking problem.

## Boards Section

This section indicates the communication boards currently used by the OSI stack with their characteristics. It is displayed only if there is no alarm. In normal operation, the state should be ON (except if explicitely disabled by the administrator, see "How to Change/Show a Subnet Entry" in the *OSI Services Reference Manual*). The same result is displayed by using the **osisnapshot** command. See the command **osisnapshot** on page 9-23.

# Chapter 6. API Trace Management

This chapter describes how to list, configure and analyze the trace of the following APIs:

- XAP API, see page 6-2,
- ROSE API, see page 6-10,
- Session API, see page 6-18.

The menus described in this chapter are displayed only if the corresponding packages are installed.

## Accessing OSI API Trace Management

### Access

From the ODIT main menu, select:

```
ODIT - OSI API Trace Management
```

**FastPath**: `odit_api`

### Overview

This menu enables the user to display current API trace levels and to change them if necessary via the following submenus:

```
ODIT -  OSI API Trace Management

XAP API
ROSE API
Session API
```

### Description

These submenus are described below in the following sections:

XAP API Trace Management on page 6-2.

ROSE API Trace Management on page 6-10.

Session API Trace Management on page 6-18.

## XAP API Trace Management

### Access

From the OSI API Trace Management menu, select:

```
XAP API
```

**Fastpath**

```
odit_xap
```

### Overview

This menu enables the user to display and configure the trace levels of the XAP API via the following submenus:

```
ODIT -  XAP API

Change/Show XAP API Trace Levels
Start/Stop Trace
Trace Analysis
```

### Description

These submenus are described in the following sections:

How to Change or Show XAP API Trace Levels on page 6-3.

How to Start or Stop the XAP API Trace on page 6-6.

How to Analyze the XAP API Trace on page 6-7.

## How to Change or Show XAP API Trace Levels

### Access

From the XAP API menu, select:

```
Change/Show XAP API Trace Levels
```

**Fastpath:** `odit_xap_level`

**Command: ositrace –ac –xap –w** *levels* **–o** *other levels*

### Overview

This menu offers three options to the user:

1. “Change/Show Trace Levels”, which leads to a screen enabling the user to display and change the trace levels specific to the XAP API by giving access to the following fields:

```
XAP API APPLICATION DEVELOPMENT TRACE
SYSTEM AND PROTOCOL ERRORS
API PRIMITIVES
API PRIMITIVES (MORE INFORMATION)
TRANSITIONS IN AUTOMATA
MESSAGES SENT FROM API TO KERNEL
MESSAGES RECEIVED BY API FROM KERNEL
INTERNAL FUNCTIONS
INTERNAL FUNCTIONS (MORE INFORMATION)
INTERNAL FUNCTIONS (XAP API FINITE STATE MACHINES)
INTERNAL FUNCTIONS (XAP API ATTRIBUTE MANAGEMENT)
INTERNAL FUNCTIONS (XAP API MEMORY MANAGEMENT)
INTERNAL FUNCTIONS (XAP API STACK INTERFACE)
XAP API INTERNAL DEBUG
OTHER DEVELOPMENT TRACE LEVELS
```

2. “Enable All Trace Levels”, to enable all trace levels,
3. “Disable All Trace Levels”, to disable all trace levels.

Once the trace levels have been configured, the XAP API trace must be started in order to record the trace. Trace levels can be reconfigured during trace recording.

### Description

This section describes the above listed fields.

The collected XAP trace levels are stored in a file located in the HOME user directory and called *.ositrace*. This default file is not used if the environment variable XAP_API_TRACE_LEVEL has been defined. The XAP API uses the trace level configuration each time an XAP API function establishing a connection is called.

Unless otherwise specified in the description of each individual field below, a trace level is enabled by setting the field to ON and disabled by setting it to OFF.

#### XAP API Application Development Trace

This trace level provides trace information for application development purposes. It is a simple and useful trace for XAP API application developers.

## System and Protocol Errors Field

This trace level provides information on the following errors and warnings:

- system errors and warnings,
- protocol errors and warnings.

This trace level is always enabled (ON).

## API Primitives Field

This trace level provides the following information on each API primitive call:

- at the beginning of the primitive
  - API primitive name,
  - input and output parameters: buffers pointed by pointer parameters are not traced.
- at the end of the primitive
  - API primitive name,
  - input and output parameters: buffers pointed by pointer parameters are not traced,
  - return code.

## API Primitives (More Information) Field

This trace level completes the API Primitives trace level. It provides the following information on each API primitive call:

- at the beginning of the primitive
  - API primitive name,
  - input and output pointer parameters: the pointed buffers are traced.
- at the end of the primitive
  - API primitive name,
  - input and output pointer parameters: the pointed buffers are traced.

## Transitions in Automata Field

This trace level provides the following information for each transition in API automata:

- automaton name
- type of event received
- initial state before event receiving
- final state after event receiving

## Messages Sent From API to Kernel Field

This trace level provides information on the stream messages sent from the API to the kernel entity.

## Messages Received by API From Kernel Field

This trace level provides information on the stream messages received by the API from the kernel entity.

## Internal Functions Field

This trace level provides the following information for each internal function call:

- at the beginning of the function:

  Function name,

  Input and output parameters: buffers pointed by pointer parameters are not traced.

- at the end of the function:

  Function name,

  Input and output parameters: buffers pointed by pointer parameters are not traced,

  Return code.

## Internal Functions (More Information) Field

This trace level completes the Internal Functions trace level. For each internal function call it provides the following information:

- at the beginning of the function:

  Function name,

  Input and output pointer parameters: pointed buffers are traced.

- at the end of the function:

  Function name,

  Input and output pointer parameters: pointed buffers are traced.

## Internal Functions (XAP API Finite State Machines)

This trace level provides information on the subset of XAP API internal functions corresponding to send and receive finite state machines.

## Internal Functions (XAP API Attribute Management)

This trace level provides information on the subset of XAP API internal functions corresponding to the management of XAP environment attributes.

## Internal Functions (XAP API Memory Management)

This trace level provides information on the subset of XAP API internal functions corresponding to memory management.

## Internal Functions (XAP API Stack Interface)

This trace level provides information on the subset of XAP API internal functions corresponding to the interface with the OSI stack upper layers.

## XAP API Internal Debug

This trace level provides information on XAP API internal debug.

## Other Development Trace Levels Field

This field can be used to enter a list of trace levels that only development teams know of. The format is a list of decimal numbers separated by spaces. The default value of the field is the list of API values of the current configuration. On a new system, the list is empty.

To enable this trace level, the user must provide a list of decimal numbers separated by spaces.

To disable this trace level, either no values are provided by the user or the existing values must be removed.

## How to Start or Stop the XAP API Trace

### Access

From the XAP API menu, select:

```
Start/Stop Trace
```

**Fastpath:** `odit_xap_st`

**Command: ositrcstart –a –xap**, or else, **ositrcstop**

### Overview

This menu enables the user to start or stop XAP API trace recording.

### Description

The user must first select “Start” in order to start the trace collection in accordance with the defined XAP levels. Then the application for which the XAP interface is to be traced must be activated. When finished, the user must select “Stop”. This not only stops the trace but also allows to start the trace analysis (see below).

**Note:** The AIX trace facility can only be started if no other trace daemon is currently running on the system.

## How to Analyze the XAP API Trace

### Access

From the XAP API menu, select:

```
Trace Analysis
```

**Fastpath:** `odit_xap_run`

**Command: ositrcrpt –a –xap** [**–fo** *filename*]

### Overview

This menu generates a trace report concerning API traces collected by the AIX trace facility. It gives access to the following dialog field:

```
FILENAME TO WRITE REPORT TO (DEFAULT IS STDOUT)
```

### Description

This section describes the above mentioned field. For a full description of the trace report produced see “Trace Report” on page A-1. Typical examples are also provided below.

#### Filename to Write Report to (Default Is Stdout) Field

This optional field enables the user to specify the file to which are saved the results of the trace analysis.

#### Trace Example 1

The following trace level is enabled (ON):

`XAP API APPLICATION DEVELOPMENT TRACE` (Level 32)

The result produced is as follows:

```
XAP API pid=16282 (level= 32): AP_OPEN
ap_open
    provider = /dev/cosp
    oflags = 0
    fd = 3

XAP API pid=16282 (level= 32): AP_INIT_ENV
ap_init_env
    fd = 3
    env_file = ./envfil_server

XAP API pid=16282 (level= 32): AP_SET_ENV
ap_set_env
    fd = 3
    attribute = AP_LIB_SEL
    value = AP_LIBVER1

XAP API pid=16282 (level= 32): AP_SET_ENV
ap_set_env
    fd = 3
    attribute = AP_BIND_PADDR
    value->p_selector = (0)
    value->s_selector = (1) |BB|
    value->t_selector = (1) |CC|
    value->n_nsaps = 1
    value->nsaps[0].nsap = (1) |0D|
    value->nsaps[0].nsap_type = AP_LOCALCOSP
```

## Trace Example 2

The following trace levels are enabled (ON):

`API PRIMITIVES` (Level 17)

`API PRIMITIVES (MORE INFORMATION)` (Level 18)

The result produced is as follows:

```
XAP API pid=9056 (level= 17): AP_OPEN >>>>>
        P1 = 0x2000082C
        P2 = 0x00000000
        P3 = 0x00000000
        P4 = 0x00000000
        P5 = 0x2FF1DD38

XAP API pid=9056 (level= 17 18): AP_OPEN
INPUT PARAM 1 :

    provider = /dev/cosp

XAP API pid=9056 (level= 17 18): AP_OPEN
INPUT PARAM 2 :

    oflags = 0

...

XAP API pid=9056 (level= 17): AP_OPEN return = 0x00000003

XAP API pid=9056 (level= 17): AP_OPEN <<<<<
```

## Trace Example 3

The following trace levels are enabled (ON):

`INTERNAL FUNCTIONS` (Level 23)

`INTERNAL FUNCTIONS (MORE INFORMATION)` (Level 24)

The result produced is as follows:

```
XAP API pid=9056 (level= 23 28): AP_GET_INSTANCE >>>>>
        P1 = 0x00000003
        P2 = 0x2FF1DCD8

XAP API pid=9056 (level= 23 28): AP_GET_INSTANCE return =
0x00000000 - error = AP_NOENV

XAP API pid=9056 (level= 23 28): AP_GET_INSTANCE <<<<<
```

XAP API trace file records are presented in the following style:

```
XAP API pid=<pid> (level= <levels>): <function> >>>>>
        P1 = <parameter>
        P2 = <parameter>
        ...
        Pn = <parameter>

XAP API pid=<pid> (level= <levels>): <function>

XAP API pid=<pid> (level= <levels>): <function> return = <return>

XAP API pid=<pid> (level= <levels>): <function> return = <return>
- error = <error>
```

```
XAP API pid=<pid> (level= <levels>): <function> <<<<<
        P1 = <parameter>
        P2 = <parameter>
        ...
        Pn = <parameter>
```

where:

<pid> is the process identifier of the XAP API calling process,

<levels> are the trace levels which have induced the current trace record,

<function>is the name of the current called function,

<return> is the return value of the current called function,

<error> is the error set by the current called function,

>>>>> indicates the beginning of the current called function,

<<<<< indicates the end of the current called function,

<parameter> is the input or output parameter of the current called function.

XAP API trace file records are presented as follows:

```
Error occurred, event ID = <event>
```

where:

<event> is an identifier which indicates the XAP code instruction where the error occurred.

Error event recording is performed each time an abnormal return code is returned to the user.

## Trace Example 4

### Error Event Recording

```
Error occurred, event ID = 07010204
```

## ROSE API Trace Management

### Access

From the OSI API Trace Management menu, select:

```
ROSE API
```

**FastPath**: `odit_rose`

### Overview

This menu enables the user to display and configure the trace levels of the ROSE API via the following submenus:

```
ODIT –  ROSE API

Change/Show ROSE API Trace Levels
Start/Stop Trace
Trace Analysis
```

### Description

These submenus are described in the following sections:

How to Change or Show ROSE API Trace Levels on page 6-11.

How to Start or Stop the ROSE API Trace on page 6-14.

How to Analyze the ROSE API Trace on page 6-15.

# How to Change or Show ROSE API Trace Levels

## Access

From the ROSE API menu, select:

```
Change/Show ROSE API Trace Levels
```

**FastPath**: `odit_rose_level`

**Command**: **ositrace –ac –rose –w** *levels* **–o** *other levels*

## Overview

This menu offers three options to the user:

1. "Change/Show Trace Levels", which leads to a screen enabling the user to display and change the trace levels specific to the ROSE API by giving access to the following fields:

   ```
   SYSTEM AND PROTOCOL ERRORS
   API PRIMITIVES
   API PRIMITIVES (MORE INFORMATION)
   TRANSITIONS IN AUTOMATA
   MESSAGES SENT FROM API TO KERNEL
   MESSAGES RECEIVED BY API FROM KERNEL
   INTERNAL FUNCTIONS
   INTERNAL FUNCTIONS (MORE INFORMATION)
   INTERNAL FUNCTIONS (SYSTEM FUNCTIONS)
   INTERNAL FUNCTIONS (MEMORY FUNCTIONS)
   INTERNAL FUNCTIONS (VERY INTERNAL FUNCTIONS)
   INTERNAL FUNCTIONS (ASN.1 FUNCTIONS)
   ROSE API INTERNAL DEBUG
   OTHER DEVELOPMENT TRACE LEVELS
   ```

2. "Enable All Trace Levels", to enable all trace levels,
3. "Disable All Trace Levels", to disable all trace levels.

Once the trace levels have been configured, the ROSE API trace must be started in order to record the trace. Trace levels can be reconfigured during trace recording by using the variable ROSE_API_TRACE_LEVEL.

## Description

This section describes the above listed fields.

The collected ROSE trace levels are stored in a file located in the HOME user directory and called *.ositrace*. This default file is not used if the environment variable ROSE_API_TRACE_LEVEL has been defined. The ROSE API uses the trace level configuration each time a ROSE API function establishing a connection is called.

Unless otherwise specified in the description of each individual field below, a trace level is enabled by setting the field to ON and disabled by setting it to OFF.

### System and Protocol Errors Field

This trace level provides information on the following errors and warnings:

- system errors and warnings,
- protocol errors and warnings.

This trace level is always enabled (ON).

## API Primitives Field

This trace level provides the following information on each API primitive call:

- at the beginning of the primitive
  - API primitive name,
  - input and output parameters: buffers pointed by pointer parameters are not traced.
- at the end of the primitive
  - API primitive name,
  - input and output parameters: buffers pointed by pointer parameters are not traced,
  - return code.

## API Primitives (More Information) Field

This trace level completes the API Primitives trace level. It provides the following information on each API primitive call:

- at the beginning of the primitive
  - API primitive name,
  - input and output pointer parameters: the pointed buffers are traced.
- at the end of the primitive
  - API primitive name,
  - input and output pointer parameters: the pointed buffers are traced.

## Transitions in Automata Field

This trace level provides the following information for each transition in API automata:

- automaton name,
- type of event received,
- initial state before event receiving,
- final state after event receiving.

## Messages Sent From API to Kernel Field

This trace level provides information on the stream messages sent from the API to the kernel entity.

## Messages Received by API From Kernel Field

This trace level provides information on the stream messages received by the API from the kernel entity.

## Internal Functions Field

This trace level provides the following information for each internal function call:

- at the beginning of the function:

  Function name,

  Input and output parameters: buffers pointed by pointer parameters are not traced.
- at the end of the function:

  Function name,

  Input and output parameters: buffers pointed by pointer parameters are not traced,

  Return code.

## Internal Functions (More Information) Field

This trace level completes the Internal Functions trace level. For each internal function call it provides the following information:

- at the beginning of the function:

  Function name,

  Input and output pointer parameters: pointed buffers are traced.

- at the end of the function:

  Function name,

  Input and output pointer parameters: pointed buffers are traced.

## Internal Functions (System Functions) Field

This trace level provides information on the subset of ROSE API functions corresponding to system calls.

## Internal Functions (Memory Functions) Field

This trace level provides information on the subset of ROSE API functions corresponding to memory management.

## Internal Functions (Very Internal Functions) Field

This trace level provides information on the subset of very basic ROSE API functions.

## Internal Functions (ASN.1 Functions) Field

This trace level provides information on the subset of ROSE API functions performing ASN.1 encoding and decoding.

## ROSE API Internal Debug Field

This trace level provides information on ROSE API internal debugging.

## Other Development Trace Levels Field

This field can be used to enter a list of trace levels that only development teams know of. The format is a list of decimal numbers separated by spaces. The default value of the field is the list of API values of the current configuration. On a new system, the list is empty.

To enable this trace level, the user must provide a list of decimal numbers separated by spaces.

To disable this trace level, either no values are provided by the user or the existing values must be removed.

## How to Start or Stop the ROSE API Trace

### Access

From the ROSE API menu, select:

```
Start/Stop Trace
```

**Fastpath:** `odit_rose_st`

**Command: ositrcstart –a –rose**, or else, **ositrcstop**

### Overview

This menu enables the user to start or stop ROSE API trace recording.

### Description

The user must first select “Start” in order to start the trace collection in accordance with the defined levels. Then the application for which the ROSE interface is to be traced must be activated. When finished, the user must select “Stop”. This not only stops the trace but also allows to start the trace analysis (see below).

**Note:** The AIX trace facility can only be started if no other trace daemon is currently running on the system.

## How to Analyze the ROSE API Trace

### Access

From the ROSE API menu, select:

```
Trace Analysis
```

**FastPath**: `odit_rose_run`

**Command**: **ositrcrpt –a –rose**

### Overview

This menu generates a trace report concerning API traces collected by the AIX trace facility. It gives access to the following dialog field:

```
FILENAME TO WRITE REPORT TO (DEFAULT IS STDOUT)
```

### Description

This section describes the above mentioned field. For a full description of the trace report produced see “Trace Report” on page A-1. Typical examples are also provided below.

#### Filename to Write Report to (Default Is Stdout) Field

This optional field enables the user to specify the file to which are saved the results of the trace analysis.

## Trace Example 1

The following trace levels are enabled:

`API PRIMITIVES` (Level 17)

`API PRIMITIVES (MORE INFORMATION)` (Level 18)

The result produced is as follows:

```
ROSE API pid=4522 (level= 17): RO_SEND >>>>>
        P1 = 0x2001D578

ROSE API pid=4522 (level= 17 18): RO_SEND
INPUT PARAM 1 :
  RO_Idu structure is :
  ---------------------
    |event      = RO_RoivReqInd
    |user_ref   = 000186A0
    |channel_id = 00000035
    |invoke_id  = 00000410
    |pci        = 00000003
    |add_info   = 00000000
    |idu        = roiv
    RO_Idu_Roiv_ReqInd substructure is :
    ------------------------------------
      |mask              =
      |operation   at @ = 0x2001B548
      |argument    at @ = 0x2001D5C8
      |op_ref            = -1
      |link_id           = 0x00000000
      |token             = Not used

      RO_OPERATION structure is :
      ---------------------------
        |type            : RO_Local
        |local value     : 1
        |class           : RO_Oclass2
        |error number    : 1
        |children number : 0

        RO_ERROR structure is :
        -----------------------
          |type            : RO_Local
          |local value     : 1

      Data substructure is :
      ----------------------
        length   : 4100
        argument :

        04 82 10 00 49 4E 56 4F 4B 45 44 41 54 41 49 4E
        56 4F 4B 45 44 41 54 41 49 4E 56 4F 4B 45 44 41
        .......................

        4B 45 44 41 54 41 49 4E 56 4F 4B 45 44 41 54 41
        49 4E 56 4F 4B 45 44 41 54 41 49 4E 56 4F 4B 45
```

## Trace Example 2

The following trace levels are enabled:

`MESSAGES RECEIVED BY API FROM KERNEL` (Level 20)

`INTERNAL FUNCTIONS` (Level 23)

`INTERNAL FUNCTIONS (SYSTEM FUNCTIONS)` (Level 27)

`ROSE API INTERNAL DEBUG` (Level 31)

The result produced is as follows:

```
ROSE API pid=4522 (level= 23 27): RO_SYST_GETMSG >>>>>
        P1 = 0x00000003

ROSE API pid=4522 (level= 20): RO_SYST_GETMSG
        Control buffer received from KERNEL:

  Stream Control Buffer :
  -----------------------
    maxlen      = 1024
    len         = 12
    buffer at @ = 0x20020C28
    buffer contents :

        00 06 03 0D 00 06 01 09 00 00 00 35

ROSE API pid=4522 (level= 20): RO_SYST_GETMSG
        Data buffer received from KERNEL:

  Stream Data Buffer :
  --------------------
    maxlen      = 32768
    len         = -1
    buffer at @ = 0x20022CC8
    buffer contents :

        empty

ROSE API pid=4522 (level= 23 27): RO_SYST_GETMSG return =
0x00000000

ROSE API pid=4522 (level= 23 27): RO_SYST_GETMSG <<<<<
        P1 = 0x2FF228B0
        P2 = 0x2FF228C0
        P3 = 0x00000001

ROSE API pid=4522 (level= 31): RO_PUT_MESSAGE
        OK_ACK  received
```

## Session API Trace Management

### Access

From the OSI API Trace Management menu, select:

```
Session API
```

**FastPath**: `odit_session`

### Overview

This menu enables the user to display and configure the trace levels of the Session API via the following submenus:

```
ODIT -  Session API

Change/Show Session API Trace Levels
Start/Stop Trace
Trace Analysis
```

### Description

These submenus are described in the following sections:

How to Change or Show Session API Trace Levels on page 6-19.

How to Start or Stop the Session API Trace on page 6-22.

How to Analyse the Session API Trace on page 6-23.

## How to Change or Show Session API Trace Levels

### Access

From the Session API menu, select:

```
Change/Show Session API Trace Levels
```

**FastPath**: `odit_session_level`

### Overview

This menu offers three options to the user:

1. “Change/Show Trace Levels”, which leads to a screen enabling the user to display and change the trace levels specific to the Session API by giving access to the following fields:

   ```
   SYSTEM AND PROTOCOL ERRORS
   API PRIMITIVES
   API PRIMITIVES (MORE INFORMATION)
   MESSAGES SENT FROM API TO KERNEL
   MESSAGES RECEIVED BY API FROM KERNEL
   INTERNAL FUNCTIONS
   INTERNAL FUNCTIONS (MORE INFORMATION)
   SYSTEM CALLS
   OTHER DEVELOPMENT TRACE LEVELS
   ```

2. “Enable All Trace Levels”, to enable all trace levels,
3. “Disable All Trace Levels”, to disable all trace levels.

Once the trace levels have been configured, the Session API trace must be started in order to record the trace. Trace levels can be reconfigured during trace recording by using the variable SESSION_API_TRACE_LEVEL.

### Description

This section describes the above listed fields.

The collected Session trace levels are stored in a file located in the HOME user directory and called *.ositrace*. This default file is not used if the environment variable SESSION_API_TRACE_LEVEL has been defined. The Session API uses the trace level configuration each time a SESSION API function establishing a connection is called.

Unless otherwise specified in the description of each individual field below, a trace level is enabled by setting the field to ON and disabled by setting it to OFF.

#### System and Protocol Errors Field

This trace level provides information on the following errors and warnings:

- system errors and warnings,
- protocol errors and warnings.

This trace level is always enabled (ON).

#### API Primitives Field

This trace level provides the following information on each API primitive call:

- at the beginning of the primitive
  - API primitive name,
  - input and output parameters: buffers pointed by pointer parameters are not traced.

- at the end of the primitive
  - API primitive name,
  - input and output parameters: buffers pointed by pointer parameters are not traced,
  - return code.

## API Primitives (More Information) Field

This trace level completes the API Primitives trace level. It provides the following information on each API primitive call:

- at the beginning of the primitive
  - API primitive name,
  - input and output pointer parameters: the pointed buffers are traced.
- at the end of the primitive
  - API primitive name,
  - input and output pointer parameters: the pointed buffers are traced.

## Messages Sent From API to Kernel Field

This trace level provides information on the stream messages sent from the API to the kernel entity.

## Messages Received by API From Kernel Field

This trace level provides information on the stream messages received by the API from the kernel entity.

## Internal Functions Field

This trace level provides the following information for each internal function call:

- at the beginning of the function:

  Function name,

  Input and output parameters: buffers pointed by pointer parameters are not traced.
- at the end of the function:

  Function name,

  Input and output parameters: buffers pointed by pointer parameters are not traced,

  Return code.

## Internal Functions (More Information) Field

This trace level completes the Internal Functions trace level. For each internal function call it provides the following information:

- at the beginning of the function:

  Function name,

  Input and output pointer parameters: pointed buffers are traced.
- at the end of the function:

  Function name,

  Input and output pointer parameters: pointed buffers are traced.

## System Calls Field

This trace level provides information on the subset of Session API function corresponding to system calls.

## Other Development Trace Levels Field

This field can be used to enter a list of trace levels that only development teams know of. The format is a list of decimal numbers separated by spaces. The default value of the field is the list of API values of the current configuration. On a new system, the list is empty.

To enable this trace level, the user must provide a list of decimal numbers separated by spaces.

To disable this trace level, either no values are provided by the user or the existing values must be removed.

## How to Start or Stop the Session API Trace

### Access

From the Session API menu, select:

```
Start/Stop Trace
```

**FastPath**: `odit_session_st`

**Command**: **ositrcstart –a –cosp**, or else, **ositrcstop**

### Overview

This menu enables the user to start or stop Session API trace recording.

### Description

The user must first select “Start” in order to start the trace collection in accordance with the defined levels. Then the application for which the Session interface is to be traced must be activated. When finished, the user must select “Stop”. This not only stops the trace but also allows to start the trace analysis (see below).

**Note:** The AIX trace facility can only be started if no other trace daemon is currently running on the system.

## How to Analyze the Session API Trace

### Access

From the Session API menu, select:

```
Trace Analysis
```

**FastPath**: `odit_session_run`

**Command**: **ositrcrpt –a –cosp**

### Overview

This menu generates a trace report concerning API traces collected by the AIX trace facility. It gives access to the following dialog field:

```
FILENAME TO WRITE REPORT TO (DEFAULT IS STDOUT)
```

### Description

This section describes the above mentioned field. For a full description of the trace report produced see “Trace Report” on page A-1. A typical example is provided below.

#### Filename to Write Report to (Default Is Stdout) Field

This optional field enables the user to specify the file to which are saved the results of the trace analysis.

## Example

The following trace levels are enabled:

`SYSTEM AND PROTOCOL ERRORS` (always ON)

`API PRIMITIVES`

The result produced is as follows:

```
OSI Traces  COSP-API:       object_name: 375C       connection_name: 10000
 >>>> Se_aDiscardAct_rq (channel_id + in_data_buf + in_lg_data_buf + in_scb
+ return_code)
    0x20000AD0      0x00000000      0x20000D90      0x20000A90
0x20000A60

OSI Traces  COSP-API:       object_name: 375C       connection_name: 10000
 parameter (channel_id)
    0x00010000

OSI Traces  COSP-API:       object_name: 375C       connection_name: 10000
 <<<< Se_aDiscardAct_rq (return_code + return)
    0x00000000      0x00000000      0x00000000      0x00000000
0x00000000

OSI Traces  COSP-API:       object_name: 375C       connection_name: 10000
 >>>> Se_Data_rq (channel_id + in_p_buffer + in_lg_buffer + in_scb +
out_scb)
    0x20000AD0      0x2FEAEA74      0x2FEAEA70      0x20000A90
0x20000DB0

OSI Traces  COSP-API:       object_name: 375C       connection_name: 10000
 >>>> Se_Data_rq (return_code)
    0x20000A60      0x00000000      0x00000000      0x00000000
0x00000000

OSI Traces  COSP-API:       object_name: 375C       connection_name: 10000
 parameter (channel_id)
    0x00010000

OSI Traces  COSP-API:       object_name: 375C       connection_name: 10000
 ERROR:S_ERROR_ACK received

OSI Traces  COSP-API:       object_name: 375C       connection_name: 10000
 <<<< Se_Data_rq (errno + return_code + return)
    0x05010033      0x08010017      0xFFFFFFFF      0x00000000
0x00000000
```

# Chapter 7. OSI Stack Layers Trace Management

This chapter describes the following functions offered by the OSI Stack internal trace management facility:

- List trace configuration, see page 7-3,
- Reset trace configuration, see page 7-6,
- Show general trace configuration, see page 7-7,
- Change or show OSI layers trace levels, see page 7-9,
- Reset the trace buffer, see page 7-51,
- Perform a trace analysis, see page 7-52,
- Save a trace to a file, see page 7-54.

## Accessing OSI Stack Layers Trace Management

### Access

From the OSI Diagnostic Interactive Tool menu, select:

```
OSI Layers Trace Management
```

**FastPath**: `odit_stack`

### Overview

This menu gives access to the OSI layers trace management operations.

```
ODIT -  OSI Layers Trace Management

List Trace Configuration
Reset Trace Configuration
General Trace Configuration
Change/Show OSI Layers Trace Levels
Reset Trace Buffer
Trace Analysis
```

### Description

This menu enables the user to list and configure OSI layers trace levels, to reset the trace buffer and to analyze the trace. See the detailed information on the operations offered by this menu in the following sections.

List Trace Configuration
: This menu displays the current trace configuration of OSI stack general parameters and OSI stack layers. See “How to List the OSI Stack Trace Configuration” on page 7-3.

Reset Trace Configuration
: This action affects previously selected trace parameters by resetting them to their default values. See “How to Reset the OSI Layer Trace Configuration” on page 7-6.

General Trace Configuration
: This menu is used to display and modify OSI stack general parameters, including ICB/UCB trace flags. See “How to Change or Show General Trace Configuration” on page 7-7.

| | |
|---|---|
| Change/Show OSI Layers Trace Levels | This menu allows to activate and deactivate OSI driver traces, OSI layer traces, ICB/UCB information for each layer. See “How to Change or Show OSI Layers Trace Levels” on page 7-9. |
| Reset Trace Buffer | This menu is used to reset the OSI stack trace buffer. See “How to Reset the Trace Buffer” on page 7-51. |
| Trace Analysis | This menu is used to perform an OSI driver trace analysis which generates either a comprehensive trace report or a summary report. See “How to Perform Trace Analysis” on page 7-52. |

## How to List the OSI Stack Trace Configuration

### Access

From the ODIT – OSI Layers Trace Management menu, select:

```
List Trace Configuration
```

**FastPath**: `odit_st_list`

**Command**: **ositrace –lc**

### Overview

This menu enables the user to display the current trace configuration of:

- General OSI parameters, i.e. ICB/UCB trace flags, truncature messages, trace buffer lengths.
- OSI stack layers.

Only the system administrator with super user authority can list all trace configurations.

### Description

This is an example of  configuration display :

```
GENERAL TRACE CONFIGURATION

OSI STACK TRACE BUFFER LENGTH : 512000 bytes
MAXIMUM TRACE MESSAGE LENGTH : 1024 bytes
ALL ICBs : NOT TRACED
ALL UCBs : NOT TRACED

KERNEL TRACE CONFIGURATION

ACSE    ERR
COPP    ERR
COSP    ERR
COTPam  ACC METH
COTPdv  ERR             CN DCN          MSG LOW
COTP    ERR             PDU             AUT
X253am  ICB
X25.3
X25.2   ICB
CLNP    ERR             GEN AUT         ICB
ES-IS   ICB
LLC
TIMER   ICB
ADMIN   ICB
MAD     DRIVER          FIFO
```

The list of mnemonics used during the display is given below.

**MNEMONICS USED FOR OSI LAYERS :**

| | |
|---|---|
| ACSE | **A**ssociation **C**ontrol **S**ervice **E**lement |
| ADMIN | **ADMIN**istration |
| CLNP | **C**onnection**L**ess **N**etwork **P**rotocol |
| COPP | **C**onnection **O**riented **P**resentation **P**rotocol |
| COSP | **C**onnection **O**riented **S**ession **P**rotocol |
| COTP | **C**onnection **O**riented **T**ransport **P**rotocol |
| COTPam | **C**onnection **O**riented **T**ransport **P**rotocol **A**ccess **M**ethod |
| COTPdv | **C**onnection **O**riented **T**ransport **P**rotocol **D**ri**V**er |
| ES-IS | **E**nd **S**ystem - **I**ntermediate **S**ystem |
| GM | Memory Manager (**G**estionnaire **M**émoire) |
| LLC | **L**ogical **L**ink **C**ontrol |
| MAD | **M**éthode **d'A**ccès **D**istribuée |
| PX25 | **P**rocess **X25** |
| TIMER | **TIMER** |
| X25.2 | **X25** level **2** |
| X25.3 | **X25** level **3** |
| X253am | **X25** **A**ccess **M**ethod |

**MNEMONICS USED FOR UPPER LAYERS TRACE LEVELS:**

| | |
|---|---|
| ERR | SYSTEM AND PROTOCOL ERRORS |
| PDU SENT | PROTOCOL DATA UNITS SENT BY THE LAYER |
| PDU REC | PROTOCOL DATA UNITS RECEIVED BY THE LAYER |
| SDU SENT LOW | SERVICE DATA UNITS SENT BY LOWER LAYER |
| SDU REC LOW | SERVICE DATA UNITS RECEIVED BY LOWER LAYER |
| SDU SENT UPP | SERVICE DATA UNITS SENT BY UPPER LAYER |
| SDU REC UPP | SERVICE DATA UNITS RECEIVED BY UPPER LAYER |
| AUT | TRANSITIONS IN AUTOMATA |
| CONF | CONFIGURATION ACTIONS |
| MGT | MANAGEMENT ACTIONS |
| FCT | INTERNAL FUNCTIONS |
| FCT MORE | INTERNAL FUNCTIONS (MORE INFORMATION) |
| ENT PT | ENTRY POINTS OF THE LAYER |
| AUT DBG | AUTOMATA DEBUG |
| STR MESS | CALL TO STREAMS (MESSAGE SENDING) |
| STR QUEUE | CALL TO STREAMS (QUEUE MANAGEMENT) |
| STR MEM | CALL TO STREAMS (MEMORY MANAGEMENT) |
| STR MASK | CALL TO STREAMS (MASK MANAGEMENT) |
| SYS MEM | CALL TO SYSTEM (MEMORY MANAGEMENT) |
| SYS TIME | CALL TO SYSTEM (TIMER MANAGEMENT) |
| SYS LCK | CALL TO SYSTEM (LOCK MANAGEMENT) |
| PERF ENT PT | PERFORMANCE (ENTRY POINTS OF THE LAYER) |
| PERF MESS | PERFORMANCE (STREAM MESSAGE HANDLING) |
| PERF TIME HAND | PERFORMANCE (TIMER HANDLERS) |
| LEVEL i | OTHER DEVELOPMENT TRACE LEVELS |

**MNEMONICS USED FOR OSI LAYERS TRACE LEVELS:**

| | |
|---|---|
| ICB | INTERNAL CONTROL BLOCK |
| UCB | USER CONTROL BLOCK |

**MNEMONICS USED FOR COTP ACCESS METHOD TRACE LEVELS:**

| | |
|---|---|
| ACC METH | ACCESS METHOD TRACE |

**MNEMONICS USED FOR COTP DRIVER TRACE LEVELS:**

| | |
|---|---|
| ERR | SYSTEM AND PROTOCOL ERRORS |
| CN DCN | CONNECTION AND DISCONNECTION |
| CN DCN MORE | CONNECTION AND DISCONNECTION (MORE INFORMATION) |
| MSG LOW | MESSAGES EXCHANGED WITH LOWER LAYER |
| MSG UPP | MESSAGES EXCHANGED WITH UPPER LAYER |
| MSG UPP MORE | MESSAGES EXCHANGED WITH UPPER LAYER (MORE INFO) |

**MNEMONICS USED FOR COTP TRACE LEVELS:**

| | |
|---|---|
| ERR | SYSTEM AND PROTOCOL ERRORS |
| CN | CONNECTION |
| DCN | DISCONNECTION REASON |
| PDU | PROTOCOL DATA UNITS SENT AND RECEIVED |
| AUT | TRANSITIONS IN AUTOMATA |
| FCT | INTERNAL FUNCTIONS |
| MAD MEM | CALL TO MAD (MEMORY MANAGEMENT) |
| DEBUG | OTHER DEVELOPEMENT TRACE LEVEL |

**MNEMONICS USED FOR CLNP TRACE LEVELS:**

| | |
|---|---|
| ERR | SYSTEM AND PROTOCOL ERRORS |
| WARN | SYSTEM AND PROTOCOL WARNINGS |
| GEN AUT | GENERAL PROTOCOL AUTOMATA |
| GEN AUT MORE | GENERAL PROTOCOL AUTOMATA (MORE INFORMATION) |
| LAN AUT | LAN INTERFACE AUTOMATA |
| LAN AUT MORE | LAN INTERFACE AUTOMATA (MORE INFORMATION) |
| WAN AUT | WAN INTERFACE AUTOMATA |
| WAN AUT MORE | WAN INTERFACE AUTOMATA (MORE INFORMATION) |

**MNEMONICS USED FOR MAD TRACE LEVELS:**

| | |
|---|---|
| ALL ICBs | ALL INTERNAL CONTROL BLOCKS |
| ALL UCBs | ALL USER CONTROL BLOCKS |
| DRIVER | DRIVER TRACE |
| SYS MEM | MEMORY TRACE |
| TIMER | TIMER TRACE |
| PERF | PERFORMANCE TRACE |
| FIFO | FIFO LIST TRACE |
| ACC METH | ACCESS METHOD |
| MAD MEM | MEMORY MANAGER TRACE |

## Possible Errors

On selecting the function “List Trace Configuration”, the following message may appear:

```
Permission denied! Super user only!
```

Super user access permission is required in order to use this function.

## How to Reset the OSI Layer Trace Configuration

### Access

From the ODIT – OSI Layers Trace Management menu, select:

```
Reset Trace Configuration
```

**FastPath**: `odit_st_rconf`

**Command**: **ositrace –rc**

### Overview

This command resets the OSI Stack trace to default values, and all OSI driver trace buffers that are dedicated to OSI Stack internal trace management. This action removes all existing traces concerning OSI driver processing, OSI layers processing, User or Internal Control Blocks and PDUs.

### Possible Errors

On selecting the function “Reset Trace Configuration”, the following message may appear:

```
An ODM system error has occurred
```

There can be two reasons for this message: either the user is not logged in as root or the ODM base has been corrupted and, therefore, the ODIT OPP needs to be re–installed.

## How to Change or Show General Trace Configuration

### Access

From the ODIT – OSI Layers Trace Management menu, select:

```
General Trace Configuration
```

**FastPath**: `odit_st_gen`

**Command**: **ositrace –gc –w** *ICB UCB BufferLength MessageLength*

### Overview

This menu function gives access to the following fields:

```
ALL INTERNAL CONTROL BLOCKS
ALL USER CONTROL BLOCKS
OSI STACK TRACE BUFFER LENGTH (IN BYTES)
MAXIMUM TRACE MESSAGE LENGTH (IN BYTES)
```

### Description

This menu function allows the administrator to display and change the following OSI stack general parameters:

- ALL Internal Control Blocks:

  This field is a switch which enables or disables the ICB trace. A List button proposes a choice between "YES" and "NO". If the value "NO" is selected, no ICBs are traced. If the value "YES" is selected, all ICBs are traced, except if the trace has been disabled at layer level.

- ALL User Control Blocks:

  This field is a switch which enables or disables the UCB trace. A List button proposes a choice between "YES" and "NO". If the value "NO" is selected, no UCBs are traced. If the value "YES" is selected, all UCBs are traced, except if the trace has been disabled at layer level.

- Length of OSI stack trace buffer (in bytes):

  This field allows to specify a decimal value in bytes for the length of the internal trace buffer. The maximum value is 1 Mbyte. The default value is 128kbytes. This circular buffer contains all the OSI layer traces.

- Maximum length of trace message (in bytes):

  This field allows to specify a decimal value in bytes for the maximum length of a trace message. The maximum value is 1024. The default value is 128. The OSI stack trace messages are truncated if their length is greater than the specified maximum value.

### Possible Errors

On selecting the function "General Trace Configuration", the following messages may appear:

```
Permission denied! Super user only!
```

Super user access permission is required in order to use this function.

```
The attribute must be within the range 131 072 -> 1 048 576
```

```
The attribute must be within the range 128 -> 1024
```

In the two above messages, the value supplied is not within the specified range.

```
Warning! ODM configuration updated but OSI stack cannot be
accessed
```

When this message appears, either the OSI stack has not been loaded or the OSI trace driver is busy (only one user access at the time is allowed).

```
   An ODM system error has occurred
```

There can be two reasons for this message: either the user is not logged in as root or the ODM base has been corrupted and, therefore, the ODIT OPP needs to be re–installed.

## How to Change or Show OSI Layers Trace Levels

### Access

From the ODIT – OSI Layers Trace Management menu, select:

```
Change/Show OSI Layers Trace Levels
```

**FastPath**: `odit_st_level`

### Overview

This menu gives access to the OSI layers trace level information.

```
ODIT – Change/Show OSI Layers Trace Levels

Change/Show ACSE Trace Levels
Change/Show COPP Trace Levels
Change/Show COSP Trace Levels
Change/Show COTP Access Method Trace Levels
Change/Show COTP Driver Trace Levels
Change/Show COTP Trace Levels
Change/Show X25.3 Access Method Trace Levels
Change/Show X25.3 Trace Levels
Change/Show X25.2 Trace Levels
Change/Show CLNP Trace Levels
Change/Show ES–IS Trace Levels
Change/Show LLC Trace Levels
Change/Show TIMER Trace Levels
Change/Show Administration Trace Levels
Change/Show MAD Trace Levels
```

### Description

Only the system administrator with super user authority can change or show the trace level of OSI layers.

Each of the above listed menu functions gives access to a number of fields which are described in the sections below.

## How to Change or Show ACSE Trace Levels

### Access

From the ODIT – Change/Show OSI Layers Trace Levels, select:

```
Change/Show ACSE Trace Levels
```

**FastPath**: `odit_st_acse`

**Command**: **ositrace –kc –acse –w** *Levels **

### Overview

ACSE stands for Association Control Service Element. It is layer 7 of the OSI model. The Change/Show ACSE Trace Levels menu allows to display and modify the trace levels which are specific to the ACSE level by giving access to the following fields.

INTERNAL CONTROL BLOCK (ICB)
SYSTEM AND PROTOCOL ERRORS
PROTOCOL DATA UNITS SENT BY THE LAYER
PROTOCOL DATA UNITS RECEIVED BY THE LAYER
SERVICE DATA UNITS SENT TO THE LOWER LAYER
SERVICE DATA UNITS RECEIVED FROM THE LOWER LAYER
SERVICE DATA UNITS SENT TO THE UPPER LAYER
SERVICE DATA UNITS RECEIVED FROM THE UPPER LAYER
TRANSITIONS IN AUTOMATA
ENTRY POINTS OF THE LAYER
INTERNAL FUNCTIONS
INTERNAL FUNCTIONS (MORE INFORMATION)
AUTOMATA DEBUG
CONFIGURATION ACTIONS
MANAGEMENT ACTIONS
CALL TO STREAMS (MESSAGE SENDING)
CALL TO STREAMS (QUEUE MANAGEMENT)
CALL TO STREAMS (MEMORY MANAGEMENT)
CALL TO STREAMS (MASK MANAGEMENT)
CALL TO SYSTEM (MEMORY MANAGEMENT)
CALL TO SYSTEM (TIMER MANAGEMENT)
CALL TO SYSTEM (LOCK MANAGEMENT)
PERFORMANCE (ENTRY POINTS OF THE LAYER)
PERFORMANCE (STREAM MESSAGE HANDLING)
PERFORMANCE (TIMER HANDLERS)
OTHER DEVELOPMENT TRACE LEVELS

### Description

This section describes the above listed fields.

Unless otherwise specified in the description of each individual field below, a trace level is enabled by setting the field to ON and disabled by setting it to OFF.

#### Internal Control Block (ICB) Field

This trace level traces ICBs which are exchanged by ACSE with the administration module (MAG).

The possible values the field can take and that are offered by the List button are:

TRACED,
NOT TRACED,
TRACED AS SOURCE,
TRACED AS DEST,
TRACED AS SOURCE & DEST.

To enable this trace level either of the above listed fields must be selected and the field ALL INTERNAL CONTROL BLOCKS (ICBS) in the General Trace Configuration dialog must be set to ON.

If the field TRACED is selected, all ICBs sent and received by the kernel entity are traced.

If the field TRACED AS SOURCE is selected, all ICBs sent by the kernel entity are traced.

If the field TRACED AS DEST is selected, all ICBs received by the kernel entity are traced.

If the field TRACED AS SOURCE & DEST is selected, ICBs are traced depending on their source or destination trace level.

To disable the ICB trace level, the field NOT TRACED must be selected.

## System and Protocol Errors Field

This trace level provides information on the following errors and warnings:

- system errors and warnings,
- protocol errors and warnings.

This trace level is always enabled (ON).

## Protocol Data Units Sent by the Layer Field

This trace level provides information on the protocol data units which are sent by the layer (hexadecimal format).

## Protocol Data Units Received by the Layer Field

This trace level provides information on the protocol data units which are received by the layer (hexadecimal format).

## Service Data Units Sent To the Lower Layer Field

This trace level provides information on the service data units which are sent to the lower interface of the layer.

## Service Data Units Received From the Lower Layer Field

This trace level provides information on the service data units which are received from the lower interface of the layer.

## Service Data Units Sent To the Upper Layer Field

This trace level provides information on the service data units which are sent to the upper interface of the layer.

## Service Data Units Received From the Upper Layer Field

This trace level provides information on the service data units which are received from the upper interface of the layer.

## Transitions in Automata Field

This trace level provides the following information for each transition in layer automata:

- automaton name,
- type of event received,
- initial state before event receiving,
- final state after event receiving.

## Entry Point of the Layer Field

This trace level provides the following information for each entry point call:

- at the beginning of the entry point:

  Entry point name,

  Input and output parameters: buffers pointed by pointer parameters are not traced.

- at the end of the entry point:

  Entry point name,

  Input and output parameters: buffers pointed by pointer parameters are not traced,

  Return code.

The entry points of the layer are the following:

Open procedure,

Close procedure,

Put procedures,

Service procedures.

## Internal Functions Field

This trace level provides the following information for each internal function call:

- at the beginning of the function:

  Function name,

  Input and output parameters: buffers pointed by pointer parameters are not traced.

- at the end of the function:

  Function name,

  Input and output parameters: buffers pointed by pointer parameters are not traced,

  Return code.

## Internal Functions (More Information) Field

This trace level completes the Internal Functions trace level. For each internal function call it provides the following information:

- at the beginning of the function:

  Function name,

  Input and output pointer parameters: pointed buffers are traced.

- at the end of the function:

  Function name,

  Input and output pointer parameters: pointed buffers are traced.

## Automata Debug Field

This trace level provides information on the layer automata at debug level.

## Configuration Actions Field

This trace level provides information on all the configuration actions which are performed in the kernel entity code.

## Management Actions Field

This trace level provides information on all management actions which are performed in the kernel entity code (except for configuration actions).

### Call to Streams (Message Sending) Field

This trace level traces calls to STREAMS that are associated with message sending. The following calls are traced:

- putnext
- qreply

To enable this trace level, set field to ON, to disable, set to OFF.

### Call to Streams (Queue Management) Field

This trace level traces calls to STREAMS that are associated with queue management. The following calls are traced:

- enableok
- flushq
- getq
- insq
- noenable
- putbq
- putq
- qenable
- rmvq

### Call to Streams (Memory Management) Field

This trace level traces calls to STREAMS that are associated with memory management. The following calls are traced:

- allocb
- bufcall
- dupb
- dupmsg
- esballoc
- freeb
- freemsg
- unbufcal

### Call to Streams (Mask Management) Field

This trace level traces calls to STREAMS that are associated with mask management. The following calls are traced:

- masking against STREAMS
- demasking against STREAMS

### Call to System (Memory Management) Field

This trace level traces calls to SYSTEM that are associated with memory management. The following calls are traced:

- buffer allocation
- buffer deallocation
- context allocation
- context deallocation

## Call to System (Timer Management) Field

This trace level traces calls to SYSTEM that are associated with timer management. The following calls are traced:

- timer activation,
- timer deactivation,
- masking against timers,
- demasking against timers.

## Call to System (Lock Management) Field

This trace level traces calls to SYSTEM that are associated with lock management. The following calls are traced:

- entering in critical section of code,
- exiting from critical section of code.

## Performance (Entry Points of the Layer) Field

This trace level is used to measure performance on layer entry points. The entry points of the layer are the following:

Open procedure,

Close procedure,

Put procedures,

Service procedures.

## Performance (Stream Message Handling) Field

This trace level is used to measure performance on functions that handle messages in and out of stream queues.

## Performance (Timer Handlers) Field

This trace level is used to measure performance on functions that handle timer interrupts. These functions are called timer handlers.

## Other Development Trace Levels Field

This field can be used to enter a list of trace levels that only development teams know of. The format is a list of decimal numbers separated by spaces. The default value of the field is the list of current kernel entity values. On a new system, the list is empty.

To enable this trace level, the user must provide a list of decimal numbers separated by spaces.

To disable this trace level, either no values are provided by the user or the existing values must be removed.

## Possible Errors

On selecting the function “Change/show ACSE Trace Levels”, the following messages may appear:

```
Permission denied! Super user only!
```

Super user access permission is required in order to use this function.

```
Warning! ODM configuration updated but OSI stack cannot be
accessed
```

When this message appears, either the OSI stack has not been loaded or the OSI trace driver is busy (only one user access at the time is allowed).

```
An ODM system error has occurred
```

There can be two reasons for this message: either the user is not logged in as root or the ODM base has been corrupted and, therefore, the ODIT OPP needs to be re–installed.

## How to Change or Show COPP Trace Levels

### Access

From the ODIT – Change/Show OSI Layers Trace Levels, select:

```
Change/Show COPP Trace Levels
```

**FastPath**: `odit_st_copp`

**Command**: **ositrace –kc –copp –w** *Levels **

### Overview

COPP stands for Connection Oriented Presentation Protocol. It is layer 6 of the OSI model. The Change/Show COPP Trace Levels menu allows to display and modify the trace levels which are specific to the COPP level by giving access to the following fields:

INTERNAL CONTROL BLOCK (ICB)
SYSTEM AND PROTOCOL ERRORS
PROTOCOL DATA UNITS SENT BY THE LAYER
PROTOCOL DATA UNITS RECEIVED BY THE LAYER
SERVICE DATA UNITS SENT TO THE LOWER LAYER
SERVICE DATA UNITS RECEIVED FROM THE LOWER LAYER
SERVICE DATA UNITS SENT TO THE UPPER LAYER
SERVICE DATA UNITS RECEIVED FROM THE UPPER LAYER
TRANSITIONS IN AUTOMATA
ENTRY POINTS OF THE LAYER
INTERNAL FUNCTIONS
INTERNAL FUNCTIONS (MORE INFORMATION)
AUTOMATA DEBUG
CONFIGURATION ACTIONS
MANAGEMENT ACTIONS
CALL TO STREAMS (MESSAGE SENDING)
CALL TO STREAMS (QUEUE MANAGEMENT)
CALL TO STREAMS (MEMORY MANAGEMENT)
CALL TO STREAMS (MASK MANAGEMENT)
CALL TO SYSTEM (MEMORY MANAGEMENT)
CALL TO SYSTEM (TIMER MANAGEMENT)
CALL TO SYSTEM (LOCK MANAGEMENT)
PERFORMANCE (ENTRY POINTS OF THE LAYER)
PERFORMANCE (STREAM MESSAGE HANDLING)
PERFORMANCE (TIMER HANDLERS)
OTHER DEVELOPMENT TRACE LEVELS

### Description

This section describes the above listed fields.

Unless otherwise specified in the description of each individual field below, a trace level is enabled by setting the field to ON and disabled by setting it to OFF.

#### Internal Control Block (ICB) Field

This trace level traces ICBs which are exchanged by the COPP layer with the administration module (MAG).

The possible values the field can take and that are offered by the List button are:

TRACED

NOT TRACED

TRACED AS SOURCE

TRACED AS DEST

TRACED AS SOURCE & DEST

To enable this trace level either of the above listed fields must be selected and the field ALL INTERNAL CONTROL BLOCKS (ICBS) in the General Trace Configuration dialog must be set to ON.

If the field TRACED is selected, all ICBs sent and received by the kernel entity are traced.

If the field TRACED AS SOURCE is selected, all ICBs sent by the kernel entity are traced.

If the field TRACED AS DEST is selected, all ICBs received by the kernel entity are traced.

If the field TRACED AS SOURCE & DEST is selected, ICBs are traced depending on their source or destination trace level.

To disable the ICB trace level, the field NOT TRACED must be selected.

## System and Protocol Errors Field

This trace level traces the following errors and warnings:

- system errors and warnings,
- protocol errors and warnings.

This trace level is always enabled (ON).

## Protocol Data Units Sent by the Layer Field

This trace level traces the protocol data units which are sent by the layer (hexadecimal format).

## Protocol Data Units Received by the Layer Field

This trace level traces the protocol data units which are received by the layer (hexadecimal format).

## Service Data Units Sent To the Lower Layer Field

This trace level traces the service data units which are sent to the lower interface of the layer.

## Service Data Units Received From the Lower Layer Field

This trace level traces the service data units which are received from the lower interface of the layer.

## Service Data Units Sent To the Upper Layer Field

This trace level traces the service data units which are sent to the upper interface of the layer.

## Service Data Units Received From the Upper Layer Field

This trace level traces the service data units which are received from the upper interface of the layer.

## Transitions in Automata Field

This trace level traces the following information for each transition in layer automata:

- automaton name
- type of event received
- initial state before event receiving
- final state after event receiving

## Entry Points of the Layer Field

This trace level traces the following information for each entry point call:

- at the beginning of the entry point:

  Entry point name,

  Input and output parameters: buffers pointed by pointer parameters are not traced.

- at the end of the entry point:

  Entry point name,

  Input and output parameters: buffers pointed by pointer parameters are not traced,

  Return code.

The entry points of the layer are the following:

  Open procedure

  Close procedure

  Put procedures

  Service procedures

## Internal Functions Field

This trace level traces the following information for each internal function call:

- at the beginning of the function:

  Function name,

  Input and output parameters: buffers pointed by pointer parameters are not traced.

- at the end of the function:

  Function name,

  Input and output parameters: buffers pointed by pointer parameters are not traced,

  Return code.

## Internal Functions (More Information) Field

This trace level completes the Internal Functions trace level. For each internal function call it traces the following information:

- at the beginning of the function:

  Function name,

  Input and output pointer parameters: pointed buffers are traced.

- at the end of the function:

  Function name,

  Input and output pointer parameters: pointed buffers are traced.

## Automata Debug Field

This trace level traces the layer automata at debug level.

To enable this trace level, set field to ON, to disable, set to OFF.

## Configuration Actions Field

This trace level traces all the configuration actions which are performed in the kernel entity code.

## Management Actions Field

This trace level traces all management actions which are performed in the kernel entity code (except for configuration actions).

## Call to Streams (Message Sending) Field

This trace level traces calls to STREAMS that are associated with message sending. The following calls are traced:

- putnext
- qreply

## Call to Streams (Queue Management) Field

This trace level traces calls to STREAMS that are associated with queue management. The following calls are traced:

- enableok
- flushq
- getq
- insq
- noenable
- putbq
- putq
- qenable
- rmvq

## Call to Streams (Memory Management) Field

This trace level traces calls to STREAMS that are associated with memory management. The following calls are traced:

- allocb
- bufcall
- dupb
- dupmsg
- esballoc
- freeb
- freemsg
- unbufcal

## Call to Streams (Mask Management) Field

This trace level traces calls to STREAMS that are associated with mask management. The following calls are traced:

- masking against STREAMS
- demasking against STREAMS

### Call to System (Memory Management) Field

This trace level traces calls to SYSTEM that are associated with memory management. The following calls are traced:

- buffer allocation
- buffer deallocation
- context allocation
- context deallocation

### Call to System (Timer Management) Field

This trace level traces calls to SYSTEM that are associated with timer management. The following calls are traced:

- timer activation
- timer deactivation
- masking against timers
- demasking against timers

### Call to System (Lock Management) Field

This trace level traces calls to SYSTEM that are associated with lock management. The following calls are traced:

- entering in critical section of code,
- exiting from critical section of code.

### Performance (Entry Points of the Layer) Field

This trace level is used to measure performance on layer entry points. The entry points of the layer are the following:

Open procedure

Close procedure

Put procedures

Service procedures

### Performance (Stream Message Handling) Field

This trace level is used to measure performance on functions that handle messages in and out of stream queues.

### Performance (Timer Handlers) Field

This trace level is used to measure performance on functions that handle timer interrupts. These functions are called timer handlers.

### Other Development Trace Levels Field

This field can be used to enter a list of trace levels that only development teams know of. The format is a list of decimal numbers separated by spaces. The default value of the field is the list of current kernel entity values. On a new system, the list is empty.

To enable this trace level, the user must provide a list of decimal numbers separated by spaces.

To disable this trace level, either no values are provided by the user or the existing values must be removed.

## Possible Errors

On selecting the function “Change/show COPP Trace Levels”, the following messages may appear:

```
Permission denied! Super user only!
```

Super user access permission is required in order to use this function.

```
Warning! ODM configuration updated but OSI stack cannot be
accessed
```

When this message appears, either the OSI stack has not been loaded or the OSI trace driver is busy (only one user access at the time is allowed).

```
An ODM system error has occurred
```

There can be two reasons for this message: either the user is not logged in as root or the ODM base has been corrupted and, therefore, the ODIT OPP needs to be re–installed.

## How to Change or Show COSP Trace Levels

### Access

From the ODIT – Change/Show OSI Layers Trace Levels, select:

```
Change/Show COSP Trace Levels
```

**FastPath**: `odit_st_cosp`

**Command**: **ositrace –kc –cosp –w** *Levels **

### Overview

COSP stands for Connection Oriented Session Protocol. It is layer 5 of the OSI model. The Change/Show COSP Trace Levels menu allows to display and modify the trace levels which are specific to the COSP level by giving access to the following fields:

INTERNAL CONTROL BLOCK (ICB)
SYSTEM AND PROTOCOL ERRORS
PROTOCOL DATA UNITS SENT BY THE LAYER
PROTOCOL DATA UNITS RECEIVED BY THE LAYER
SERVICE DATA UNITS SENT TO THE LOWER LAYER
SERVICE DATA UNITS RECEIVED FROM THE LOWER LAYER
SERVICE DATA UNITS SENT TO THE UPPER LAYER
SERVICE DATA UNITS RECEIVED FROM THE UPPER LAYER
TRANSITIONS IN AUTOMATA
ENTRY POINTS OF THE LAYER
INTERNAL FUNCTIONS
AUTOMATA DEBUG
MANAGEMENT ACTIONS
CALL TO STREAMS (MESSAGE SENDING)
CALL TO STREAMS (QUEUE MANAGEMENT)
CALL TO STREAMS (MEMORY MANAGEMENT)
CALL TO STREAMS (MASK MANAGEMENT)
CALL TO SYSTEM (MEMORY MANAGEMENT)
CALL TO SYSTEM (LOCK MANAGEMENT)
OTHER DEVELOPMENT TRACE LEVELS

### Description

This section describes the above listed fields.

Unless otherwise specified in the description of each individual field below, a trace level is enabled by setting the field to ON and disabled by setting it to OFF.

#### Internal Control Block (ICB) Field

This trace level traces ICBs which are exchanged by the COSP layer with the administration module (MAG).

The possible values the field can take and that are offered by the List button are:

TRACED

NOT TRACED

TRACED AS SOURCE

TRACED AS DEST

TRACED AS SOURCE & DEST

To enable this trace level either of the above listed fields must be selected and the field ALL INTERNAL CONTROL BLOCKS (ICBS) in the General Trace Configuration dialog must be set to ON.

If the field TRACED is selected, all ICBs sent and received by the kernel entity are traced.

If the field TRACED AS SOURCE is selected, all ICBs sent by the kernel entity are traced.

If the field TRACED AS DEST is selected, all ICBs received by the kernel entity are traced.

If the field TRACED AS SOURCE & DEST is selected, ICBs are traced depending on their source or destination trace level.

To disable the ICB trace level, the field NOT TRACED must be selected.

## System and Protocol Errors Field

This trace level traces the following errors and warnings:

- system errors and warnings,
- protocol errors and warnings,
- user errors and warnings.

This trace level is always enabled (ON).

## Protocol Data Units Sent by the Layer Field

This trace level traces the protocol data units which are sent by the layer (hexadecimal format).

## Protocol Data Units Received by the Layer Field

This trace level traces the protocol data units which are received by the layer (hexadecimal format).

## Service Data Units Sent To the Lower Layer Field

This trace level traces the service requests and responses sent to the transport provider. In the case of data requests, the field “Protocol Data Units Sent by the Layer” must be set to ON in order to trace the data.

## Service Data Units Received From the Lower Layer Field

This trace level traces the service indications and acknowledgements received from the transport provider. In the case of a data indication, the field “Protocol Data Units Received by the Layer” must be sent to ON in order to trace the data contents.

## Service Data Units Sent To the Upper Layer Field

This trace level traces the type of the service indications confirmation or acknowledgement that are sent to the session user. To trace the contents of these messages, the field “Call to Stream (Message Sending)” must be set to ON.

## Service Data Units Received From the Upper Layer Field

This trace level traces the service data units which are received from the upper interface of the session layer.

## Transitions in Automata Field

This trace level traces state changes in the session provider (previous state – new state).

## Entry Points of the Layer Field

This trace level traces the following information for each entry point call:

- at the beginning of the entry point:

  Entry point name,

  Input and output parameters: buffers pointed by pointer parameters are not traced.

- at the end of the entry point:

  Entry point name,

  Return code.

The entry points of the layer are the following:

Open procedure

Close procedure

Put procedures

Service procedures

## Internal Functions Field

This trace level traces the following information for each internal function call:

- at the beginning of the function:

  Function name,

  Input and output parameters: buffers pointed by pointer parameters are not traced.

- at the end of the function:

  Function name,

  Return code.

## Automata Debug Field

This trace level traces the layer automata at debug level.

## Management Actions Field

This trace level traces all management actions which are performed in the kernel entity code.

## Call to Streams (Message Sending) Field

This trace level traces calls to STREAMS that are associated with message sending. The following calls are traced:

- putnext
- qreply

## Call to Streams (Queue Management) Field

This trace level traces calls to STREAMS that are associated with queue management. The following calls are traced:

- enableok
- flushq
- getq
- insq
- noenable
- putbq
- putq
- qenable
- rmvq

## Call to Streams (Memory Management) Field

This trace level traces calls to STREAMS that are associated with memory management. The following calls are traced:

- allocb
- bufcall
- dupb
- dupmsg
- esballoc
- freeb
- freemsg
- unbufcal

## Call to Streams (Mask Management) Field

This trace level traces calls to STREAMS that are associated with mask management. The following calls are traced:

- masking against STREAMS
- demasking against STREAMS

## Call to System (Memory Management) Field

This trace level traces calls to SYSTEM that are associated with memory management. The following calls are traced:

- buffer allocation
- buffer deallocation
- context allocation
- context deallocation

## Call to System (Lock Management) Field

This trace level traces calls to SYSTEM that are associated with lock management. The following calls are traced:

- entering in critical section of code,
- exiting from critical section of code.

## Other Development Trace Levels Field

This field can be used to enter a list of trace levels that only development teams know of. The format is a list of decimal numbers separated by spaces. The default value of the field is the list of current kernel entity values. On a new system, the list is empty.

To enable this trace level, the user must provide a list of decimal numbers separated by spaces.

To disable this trace level, either no values are provided by the user or the existing values must be removed.

## Possible Errors

On selecting the function “Change/show COSP Trace Levels”, the following messages may appear:

```
Permission denied! Super user only!
```

Super user access permission is required in order to use this function.

```
Warning! ODM configuration updated but OSI stack cannot be
accessed
```

When this message appears, either the OSI stack has not been loaded or the OSI trace driver is busy (only one user access at the time is allowed).

```
An ODM system error has occurred
```

There can be two reasons for this message: either the user is not logged in as root or the ODM base has been corrupted and, therefore, the ODIT OPP needs to be re–installed.

## How to Change or Show COTP Access Method Trace Levels

### Access

From the ODIT – Change/Show OSI Layers Trace Levels, select:

```
Change/Show COTP Access Method Trace Levels
```

**FastPath**: `odit_st_mcotp`

**Command**: **ositrace –kc –cotpam –w** *Levels **

### Overview

The COTP Access Method is used in some applications based on the OSI COTP layer (e.g. TPAD on LAN). The Change/Show COTP Access Method Trace Levels menu allows to display and modify the trace levels which are specific to the COTP Access Method by giving access to the following fields:

ACCESS METHOD TRACE
USER CONTROL BLOCK (UCB)
INTERNAL CONTROL BLOCK (ICB)

### Description

This section describes the above listed fields.

Unless otherwise specified in the description of each individual field below, a trace level is enabled by setting the field to ON and disabled by setting it to OFF.

#### Access Method Trace Field

This trace level traces internal processing of the COTP Access Method.

The default value is the current value of the kernel entity. On a new system, the value is set to OFF.

#### User Control Block (UCB) Field

This trace level traces UCBs which are sent and received by the OSI layer entity.

The default value is the current value of the kernel entity. On a new system, the value is set to OFF.

#### Internal Control Block (ICB) Field

This trace level traces ICBs which are exchanged by the COTP layer with the layer directly below and the layer directly above it.

The possible values the field can take and that are offered by the List button are:

TRACED

NOT TRACED

TRACED AS SOURCE

TRACED AS DEST

TRACED AS SOURCE & DEST

To enable this trace level either of the above listed fields must be selected and the field ALL INTERNAL CONTROL BLOCKS (ICBS) in the General Trace Configuration dialog must be set to ON.

If the field TRACED is selected, all ICBs sent and received by the kernel entity are traced.

If the field TRACED AS SOURCE is selected, all ICBs sent by the kernel entity are traced.

If the field TRACED AS DEST is selected, all ICBs received by the kernel entity are traced.

If the field TRACED AS SOURCE & DEST is selected, ICBs are traced depending on their source or destination trace level.

To disable the ICB trace level, the field NOT TRACED must be selected.

## Possible Errors

On selecting the function “Change/Show COTP Access Method Trace Levels”, the following messages may appear:

```
Permission denied! Super user only!
```

Super user access permission is required in order to use this function.

```
Warning! ODM configuration updated but OSI stack cannot be
accessed
```

When this message appears, either the OSI stack has not been loaded or the OSI trace driver is busy (only one user access at the time is allowed).

```
An ODM system error has occurred
```

There can be two reasons for this message: either the user is not logged in as root or the ODM base has been corrupted and, therefore, the ODIT OPP needs to be re–installed.

## How to Change or Show COTP Driver Trace Levels

### Access

From the ODIT – Change/Show OSI Layers Trace Levels, select:

```
Change/Show COTP Driver Trace Levels
```

**FastPath**: `odit_st_dcotp`

**Command**: **ositrace –kc –cotpdv –w** *Levels **

### Overview

The COTP Driver allows a stream module to access the OSI COTP layer via the TPI interface. The Change/Show COTP Driver Trace Levels menu allows to display and modify the trace levels which are specific to the COTP Driver by giving access to the following fields:

SYSTEM AND PROTOCOL ERRORS
CONNECTION AND DISCONNECTION
CONNECTION AND DISCONNECTION (MORE INFORMATION)
MESSAGES EXCHANGED WITH LOWER LAYER
MESSAGES EXCHANGED WITH UPPER LAYER
MESSAGES EXCHANGED WITH UPPER LAYER (MORE INFO)
INTERNAL CONTROL BLOCK (ICB)

### Description

This section describes the above listed fields.

Unless otherwise specified in the description of each individual field below, a trace level is enabled by setting the field to ON and disabled by setting it to OFF.

#### System and Protocol Errors Field

This trace level traces system and protocol errors.

#### Connection and Disconnection Field

This trace level provides basic information concerning connection and disconnection management for the COTP driver.

#### Connection and Disconnection (More Information) Field

This trace level provides detailed information concerning connection and disconnection management for the COTP driver.

#### Data Processing With Lower Layer Field

This trace level provides information on the ICBs exchanged between the COTP driver and the COTP layer.

#### Data Processing With Upper Layer Field

This trace level provides sample traces concerning data transfer management by the COTP driver.

#### Data Processing With Upper Layer (More Info) Field

This trace level provides detailed traces concerning data transfer management by the COTP driver.

#### Internal Control Block (ICB) Field

This trace level traces ICBs which are exchanged by the COTP layer with the layer directly below and the layer directly above it.

The possible values the field can take and that are offered by the List button are:

TRACED

NOT TRACED

TRACED AS SOURCE

TRACED AS DEST

TRACED AS SOURCE & DEST

To enable this trace level either of the above listed fields must be selected and the field ALL INTERNAL CONTROL BLOCKS (ICBS) in the General Trace Configuration dialog must be set to ON.

If the field TRACED is selected, all ICBs sent and received by the kernel entity are traced.

If the field TRACED AS SOURCE is selected, all ICBs sent by the kernel entity are traced.

If the field TRACED AS DEST is selected, all ICBs received by the kernel entity are traced.

If the field TRACED AS SOURCE & DEST is selected, ICBs are traced depending on their source or destination trace level.

To disable the ICB trace level, the field NOT TRACED must be selected.

## Possible Errors

On selecting the function “Change/show COTP Driver Trace Levels”, the following messages may appear:

```
Permission denied! Super user only!
```

Super user access permission is required in order to use this function.

```
Warning! ODM configuration updated but OSI stack cannot be
accessed
```

When this message appears, either the OSI stack has not been loaded or the OSI trace driver is busy (only one user access at the time is allowed).

```
An ODM system error has occurred
```

There can be two reasons for this message: either the user is not logged in as root or the ODM base has been corrupted and, therefore, the ODIT OPP needs to be re–installed.

## How to Change or Show COTP Trace Levels

### Access

From the ODIT – Change/Show OSI Layers Trace Levels, select:

```
Change/Show COTP Trace Levels
```

**FastPath**: `odit_st_cotp`

**Command**: **ositrace –kc –cotp –w** *Levels **

### Overview

COTP stands for Connection Oriented Transport Protocol. It is layer 4 of the OSI model. The Change/Show COTP Trace Levels menu allows to display and modify the trace levels which are specific to the COTP level by giving access to the following fields:

SYSTEM AND PROTOCOL ERRORS
CONNECTION
DISCONNECTION REASON
PROTOCOL DATA UNITS SENT AND RECEIVED
TRANSITIONS IN AUTOMATA
INTERNAL FUNCTIONS
CALL TO MAD (MEMORY MANAGEMENT)
OTHER DEVELOPMENT TRACE LEVEL
INTERNAL CONTROL BLOCK (ICB)

### Description

This section describes the above listed fields.

Unless otherwise specified in the description of each individual field below, a trace level is enabled by setting the field to ON and disabled by setting it to OFF.

#### System and Protocol Errors Field

This trace level traces system and protocol errors.

#### Connection Field

This trace level traces connection information (site table content).

#### Disconnection Reason Field

This trace level traces the cause of COTP disconnections.

#### Protocol Data Units Sent and Received Field

This trace level provides information on protocol data units sent and received by the layer.

#### Transitions in Automata Field

This trace level provides information on each state transition in layer automata.

#### Internal Functions Field

This trace level provides information on each internal function call.

#### Call to MAD (Memory Management) Field

This trace level provides information on all MAD memory management actions.

#### Other Development Trace Level Field

This trace level can be used to trace other kinds of information required by development teams for COTP layer debugging.

### Internal Control Block (ICB) Field

This trace level traces ICBs which are exchanged by the COTP layer with the layer directly below and the layer directly above it.

The possible values the field can take and that are offered by the List button are:

TRACED

NOT TRACED

TRACED AS SOURCE

TRACED AS DEST

TRACED AS SOURCE & DEST

To enable this trace level either of the above listed fields must be selected and the field ALL INTERNAL CONTROL BLOCKS (ICBS) in the General Trace Configuration dialog must be set to ON.

If the field TRACED is selected, all ICBs sent and received by the kernel entity are traced.

If the field TRACED AS SOURCE is selected, all ICBs sent by the kernel entity are traced.

If the field TRACED AS DEST is selected, all ICBs received by the kernel entity are traced.

If the field TRACED AS SOURCE & DEST is selected, ICBs are traced depending on their source or destination trace level.

To disable the ICB trace level, the field NOT TRACED must be selected.

## Possible Errors

On selecting the function “Change/show COTP Trace Levels”, the following messages may appear:

```
Permission denied! Super user only!
```

Super user access permission is required in order to use this function.

```
Warning! ODM configuration updated but OSI stack cannot be
accessed
```

When this message appears, either the OSI stack has not been loaded or the OSI trace driver is busy (only one user access at the time is allowed).

```
An ODM system error has occurred
```

There can be two reasons for this message: either the user is not logged in as root or the ODM base has been corrupted and, therefore, the ODIT OPP needs to be re–installed.

## How to Change or Show X25.3 Access Method Trace Levels

### Access

From the ODIT – Change/Show OSI Layers Trace Levels, select:

```
Change/Show X25.3 Access Method Trace Levels
```

**FastPath**: `odit_st_mx253`

**Command**: **ositrace –kc –x253am –w** *Levels **

### Overview

The X25.3 Access Method can be used to develop applications based on the OSI X25.3 layer. The Change/Show X25.3 Access Method Trace Levels menu allows to display and modify the trace levels which are specific to the X25.3 Access Method by giving access to the following fields:

USER CONTROL BLOCK (UCB)
INTERNAL CONTROL BLOCK (ICB)

### Description

This section describes the above listed fields.

#### User Control Block (UCB) Field

This trace level traces UCBs which are sent and received by the OSI layer entity.

To enable this trace level, set field to ON, to disable, set to OFF.

The default value is the current value of the kernel entity. On a new system, the value is set to OFF.

#### Internal Control Block (ICB) Field

This trace level traces ICBs which are exchanged by the COTP layer with the layer directly below and the layer directly above it.

The possible values the field can take and that are offered by the List button are:

TRACED

NOT TRACED

TRACED AS SOURCE

TRACED AS DEST

TRACED AS SOURCE & DEST

To enable this trace level either of the above listed fields must be selected and the field ALL INTERNAL CONTROL BLOCKS (ICBS) in the General Trace Configuration dialog must be set to ON.

If the field TRACED is selected, all ICBs sent and received by the kernel entity are traced.

If the field TRACED AS SOURCE is selected, all ICBs sent by the kernel entity are traced.

If the field TRACED AS DEST is selected, all ICBs received by the kernel entity are traced.

If the field TRACED AS SOURCE & DEST is selected, ICBs are traced depending on their source or destination trace level.

To disable the ICB trace level, the field NOT TRACED must be selected.

## Possible Errors

On selecting the function “Change/show X25.3 Access Method Trace Levels”, the following messages may appear:

```
Permission denied! Super user only!
```

Super user access permission is required in order to use this function.

```
Warning! ODM configuration updated but OSI stack cannot be
accessed
```

When this message appears, either the OSI stack has not been loaded or the OSI trace driver is busy (only one user access at the time is allowed).

```
An ODM system error has occurred
```

There can be two reasons for this message: either the user is not logged in as root or the ODM base has been corrupted and, therefore, the ODIT OPP needs to be re–installed.

## How to Change or Show X25.3 Trace Levels

### Access

From the ODIT – Change/Show OSI Layers Trace Levels, select:

```
Change/Show X25.3 Trace Levels
```

**FastPath**: `odit_st_x253`

**Command**: **ositrace –kc –x253 –w** *Levels **

### Overview

X25.3 belongs to layer 3 in the OSI model. The Change/Show X25.3 Trace Levels menu allows to display and modify the trace levels which are specific to the X25.3 layer by giving access to the following field:

INTERNAL CONTROL BLOCK (ICB)

### Description

This section describes the above mentioned field.

#### Internal Control Block (ICB) Field

This trace level traces ICBs which are exchanged by the X25.3 layer with the layer directly below and the layer directly above it.

The possible values the field can take and that are offered by the List button are:

TRACED

NOT TRACED

TRACED AS SOURCE

TRACED AS DEST

TRACED AS SOURCE & DEST

To enable this trace level either of the above listed fields must be selected and the field ALL INTERNAL CONTROL BLOCKS (ICBS) in the General Trace Configuration dialog must be set to ON.

If the field TRACED is selected, all ICBs sent and received by the kernel entity are traced.

If the field TRACED AS SOURCE is selected, all ICBs sent by the kernel entity are traced.

If the field TRACED AS DEST is selected, all ICBs received by the kernel entity are traced.

If the field TRACED AS SOURCE & DEST is selected, ICBs are traced depending on their source or destination trace level.

To disable the ICB trace level, the field NOT TRACED must be selected.

## Possible Errors

On selecting the function “Change/show X25.3 Trace Levels”, the following messages may appear:

```
Permission denied! Super user only!
```

Super user access permission is required in order to use this function.

```
Warning! ODM configuration updated but OSI stack cannot be
accessed
```

When this message appears, either the OSI stack has not been loaded or the OSI trace driver is busy (only one user access at the time is allowed).

```
An ODM system error has occurred
```

There can be two reasons for this message: either the user is not logged in as root or the ODM base has been corrupted and, therefore, the ODIT OPP needs to be re–installed.

## How to Change or Show X25.2 Trace Levels

### Access

From the ODIT – Change/Show OSI Layers Trace Levels, select:

```
Change/Show X25.2 Trace Levels
```

**FastPath**: `odit_st_x252`

**Command**: **ositrace –kc –x252 –w** *Levels **

### Overview

X25.2 belongs to layer 2 in the OSI model. The Change/Show X25.2 Trace Levels menu allows to display and modify the trace levels which are specific to the X25.2 layer by giving access to the following field:

INTERNAL CONTROL BLOCK (ICB)

### Description

This section describes the above mentioned field.

#### Internal Control Block (ICB) Field

This trace level traces ICBs which are exchanged by the X25.2 layer with the layer directly below and the layer directly above it.

The possible values the field can take and that are offered by the List button are:

TRACED

NOT TRACED

TRACED AS SOURCE

TRACED AS DEST

TRACED AS SOURCE & DEST

To enable this trace level either of the above listed fields must be selected and the field ALL INTERNAL CONTROL BLOCKS (ICBS) in the General Trace Configuration dialog must be set to ON.

If the field TRACED is selected, all ICBs sent and received by the kernel entity are traced.

If the field TRACED AS SOURCE is selected, all ICBs sent by the kernel entity are traced.

If the field TRACED AS DEST is selected, all ICBs received by the kernel entity are traced.

If the field TRACED AS SOURCE & DEST is selected, ICBs are traced depending on their source or destination trace level.

To disable the ICB trace level, the field NOT TRACED must be selected.

## Possible Errors

On selecting the function “Change/Show X25.2 Trace Levels”, the following messages may appear:

```
Permission denied! Super user only!
```

Super user access permission is required in order to use this function.

```
Warning! ODM configuration updated but OSI stack cannot be
accessed
```

When this message appears, either the OSI stack has not been loaded or the OSI trace driver is busy (only one user access at the time is allowed).

```
An ODM system error has occurred
```

There can be two reasons for this message: either the user is not logged in as root or the ODM base has been corrupted and, therefore, the ODIT OPP needs to be re–installed.

## How to Change or Show CLNP Trace Levels

### Access

From the ODIT – Change/Show OSI Layers Trace Levels, select:

```
Change/Show CLNP Trace Levels
```

**FastPath**: `odit_st_clnp`

**Command**: **ositrace –kc –clnp –w** *Levels **

### Overview

CLNP stands for ConnectionLess Network Protocol. It is layer 3 in the OSI model. The Change/Show CLNP Trace Levels menu allows to display and modify the trace levels which are specific to the CLNP layer by giving access to the following fields:

SYSTEM AND PROTOCOL ERRORS
SYSTEM AND PROTOCOL WARNINGS
GENERAL PROTOCOL AUTOMATA
GENERAL PROTOCOL AUTOMATA (MORE INFORMATION)
LAN INTERFACE AUTOMATA
LAN INTERFACE AUTOMATA (MORE INFORMATION)
WAN INTERFACE AUTOMATA
WAN INTERFACE AUTOMATA (MORE INFORMATION)
INTERNAL CONTROL BLOCK (ICB)

### Description

This section describes the above listed fields.

Unless otherwise specified in the description of each individual field below, a trace level is enabled by setting the field to ON and disabled by setting it to OFF.

#### System and Protocol Errors Field

This trace level traces system and protocol errors.

#### System and Protocol Warnings Field

This trace level traces system and protocol warnings.

#### General Protocol Automata Field

This trace level provides information on each transition in general protocol automata.

#### General Protocol Automata (More Information) Field

This trace level traces the general protocol automata at debug level.

#### LAN Interface Automata Field

This trace level provides information on each transition in LAN interface automata.

#### LAN Interface Automata (More Information) Field

This trace level traces LAN interface automata at debug level.

#### WAN Interface Automata Field

This trace level provides information on each transition in WAN interface automata.

#### WAN Interface Automata (More Information) Field

This trace level traces WAN interface automata at debug level.

### Internal Control Block (ICB) Field

This trace level traces ICBs which are exchanged by the CLNP layer with the layer directly below and the layer directly above it.

The possible values the field can take and that are offered by the List button are:

TRACED

NOT TRACED

TRACED AS SOURCE

TRACED AS DEST

TRACED AS SOURCE & DEST

To enable this trace level either of the above listed fields must be selected and the field ALL INTERNAL CONTROL BLOCKS (ICBS) in the General Trace Configuration dialog must be set to ON.

If the field TRACED is selected, all ICBs sent and received by the kernel entity are traced.

If the field TRACED AS SOURCE is selected, all ICBs sent by the kernel entity are traced.

If the field TRACED AS DEST is selected, all ICBs received by the kernel entity are traced.

If the field TRACED AS SOURCE & DEST is selected, ICBs are traced depending on their source or destination trace level.

To disable the ICB trace level, the field NOT TRACED must be selected.

## Possible Errors

On selecting the function “Change/Show CLNP Trace Levels”, the following messages may appear:

```
Permission denied! Super user only!
```

Super user access permission is required in order to use this function.

```
Warning! ODM configuration updated but OSI stack cannot be
accessed
```

When this message appears, either the OSI stack has not been loaded or the OSI trace driver is busy (only one user access at the time is allowed).

```
An ODM system error has occurred
```

There can be two reasons for this message: either the user is not logged in as root or the ODM base has been corrupted and, therefore, the ODIT OPP needs to be re–installed.

## How to Change or Show ES–IS Trace Levels

### Access

From the ODIT – Change/Show OSI Layers Trace Levels, select:

```
Change/Show ES-IS Trace Levels
```

**FastPath**: `odit_st_esis`

**Command**: **ositrace –kc –esis –w** *Levels **

### Overview

ES–IS is the OSI 9542 protocol. The Change/Show ES–IS Trace Levels menu allows to display and modify the trace levels which are specific to the ES–IS layer by giving access to the following field:

INTERNAL CONTROL BLOCK (ICB)

### Description

This section describes the above mentioned field.

#### Internal Control Block (ICB) Field

This trace level traces ICBs which are exchanged by the ES–IS layer with the layer directly below and the layer directly above it.

The possible values the field can take and that are offered by the List button are:

TRACED

NOT TRACED

TRACED AS SOURCE

TRACED AS DEST

TRACED AS SOURCE & DEST

To enable this trace level either of the above listed fields must be selected and the field ALL INTERNAL CONTROL BLOCKS (ICBS) in the General Trace Configuration dialog must be set to ON.

If the field TRACED is selected, all ICBs sent and received by the kernel entity are traced.

If the field TRACED AS SOURCE is selected, all ICBs sent by the kernel entity are traced.

If the field TRACED AS DEST is selected, all ICBs received by the kernel entity are traced.

If the field TRACED AS SOURCE & DEST is selected, ICBs are traced depending on their source or destination trace level.

To disable the ICB trace level, the field NOT TRACED must be selected.

## Possible Errors

On selecting the function “Change/Show ES–IS Trace Levels”, the following messages may appear:

```
Permission denied! Super user only!
```

Super user access permission is required in order to use this function.

```
Warning! ODM configuration updated but OSI stack cannot be
accessed
```

When this message appears, either the OSI stack has not been loaded or the OSI trace driver is busy (only one user access at the time is allowed).

```
An ODM system error has occurred
```

There can be two reasons for this message: either the user is not logged in as root or the ODM base has been corrupted and, therefore, the ODIT OPP needs to be re–installed.

## How to Change or Show LLC Trace Levels

### Access

From the ODIT – Change/Show OSI Layers Trace Levels, select:

```
Change/Show LLC Trace Levels
```

**FastPath**: `odit_st_llc`

**Command**: **ositrace –kc –llc1 –w** *Levels **

### Overview

LLC stands for Logical Link Control. It is the top sub–layer in the data link layer (layer 2 of the OSI model). The Change/Show LLC Trace Levels menu allows to display and modify the trace levels which are specific to the LLC sub–layer by giving access to the following field:

INTERNAL CONTROL BLOCK (ICB)

### Description

This section describes the above mentioned field.

#### Internal Control Block (ICB) Field

This trace level traces ICBs which are exchanged by the LLC sub–layer with the layer directly below and the layer directly above it.

The possible values the field can take and that are offered by the List button are:

TRACED

NOT TRACED

TRACED AS SOURCE

TRACED AS DEST

TRACED AS SOURCE & DEST

To enable this trace level either of the above listed fields must be selected and the field ALL INTERNAL CONTROL BLOCKS (ICBS) in the General Trace Configuration dialog must be set to ON.

If the field TRACED is selected, all ICBs sent and received by the kernel entity are traced.

If the field TRACED AS SOURCE is selected, all ICBs sent by the kernel entity are traced.

If the field TRACED AS DEST is selected, all ICBs received by the kernel entity are traced.

If the field TRACED AS SOURCE & DEST is selected, ICBs are traced depending on their source or destination trace level.

To disable the ICB trace level, the field NOT TRACED must be selected.

## Possible Errors

On selecting the function “Change/Show LLC Trace Levels”, the following messages may appear:

```
Permission denied! Super user only!
```

Super user access permission is required in order to use this function.

```
Warning! ODM configuration updated but OSI stack cannot be
accessed
```

When this message appears, either the OSI stack has not been loaded or the OSI trace driver is busy (only one user access at the time is allowed).

```
An ODM system error has occurred
```

There can be two reasons for this message: either the user is not logged in as root or the ODM base has been corrupted and, therefore, the ODIT OPP needs to be re–installed.

## How to Change or Show TIMER Trace Levels

### Access

From the ODIT – Change/Show OSI Layers Trace Levels, select:

```
Change/Show TIMER Trace Levels
```

**FastPath**: `odit_st_timer`

**Command**: **ositrace –kc –timer –w** *Levels **

### Overview

The Change/Show TIMER Trace Levels menu allows to display and modify the trace levels which are specific to the Timer kernel entity by giving access to the following field:

INTERNAL CONTROL BLOCK (ICB)

### Description

This section describes the above mentioned field.

#### Internal Control Block (ICB) Field

This trace level traces ICBs which are exchanged by the Timer sub–layer with the layer directly below and the layer directly above it.

The possible values the field can take and that are offered by the List button are:

TRACED

NOT TRACED

TRACED AS SOURCE

TRACED AS DEST

TRACED AS SOURCE & DEST

To enable this trace level either of the above listed fields must be selected and the field ALL INTERNAL CONTROL BLOCKS (ICBS) in the General Trace Configuration dialog must be set to ON.

If the field TRACED is selected, all ICBs sent and received by the kernel entity are traced.

If the field TRACED AS SOURCE is selected, all ICBs sent by the kernel entity are traced.

If the field TRACED AS DEST is selected, all ICBs received by the kernel entity are traced.

If the field TRACED AS SOURCE & DEST is selected, ICBs are traced depending on their source or destination trace level.

To disable the ICB trace level, the field NOT TRACED must be selected.

## Possible Errors

On selecting the function “Change/Show TIMER Trace Levels”, the following messages may appear:

```
Permission denied! Super user only!
```

Super user access permission is required in order to use this function.

```
Warning! ODM configuration updated but OSI stack cannot be
accessed
```

When this message appears, either the OSI stack has not been loaded or the OSI trace driver is busy (only one user access at the time is allowed).

```
An ODM system error has occurred
```

There can be two reasons for this message: either the user is not logged in as root or the ODM base has been corrupted and, therefore, the ODIT OPP needs to be re–installed.

## How to Change or Show Administration Trace Levels

### Access

From the ODIT – Change/Show OSI Layers Trace Levels, select:

```
Change/show Administration Trace Levels
```

**FastPath**: `odit_st_adm`

**Command**: **ositrace –kc –adm –w** *Levels **

### Overview

The Change/Show Administration Trace Levels menu allows to display and modify the trace levels which are specific to the Administration kernel entity by giving access to the following fields:

USER CONTROL BLOCK (UCB)
INTERNAL CONTROL BLOCK (ICB)

### Description

This section describes the above listed fields.

#### User Control Block (UCB) Field

This trace level traces UCBs which are sent and received by the OSI layer entity.

To enable this trace level, set field to ON, to disable, set to OFF.

The default value is the current value of the kernel entity. On a new system, the value is set to OFF.

#### Internal Control Block (ICB) Field

This trace level traces ICBs which are exchanged by the Timer sub–layer with the layer directly below and the layer directly above it.

The possible values the field can take and that are offered by the List button are:

TRACED

NOT TRACED

TRACED AS SOURCE

TRACED AS DEST

TRACED AS SOURCE & DEST

To enable this trace level either of the above listed fields must be selected and the field ALL INTERNAL CONTROL BLOCKS (ICBS) in the General Trace Configuration dialog must be set to ON.

If the field TRACED is selected, all ICBs sent and received by the kernel entity are traced.

If the field TRACED AS SOURCE is selected, all ICBs sent by the kernel entity are traced.

If the field TRACED AS DEST is selected, all ICBs received by the kernel entity are traced.

If the field TRACED AS SOURCE & DEST is selected, ICBs are traced depending on their source or destination trace level.

To disable the ICB trace level, the field NOT TRACED must be selected.

## Possible Errors

On selecting the function “Change/Show Administration Trace Levels”, the following messages may appear:

```
Permission denied! Super user only!
```

Super user access permission is required in order to use this function.

```
Warning! ODM configuration updated but OSI stack cannot be
accessed
```

When this message appears, either the OSI stack has not been loaded or the OSI trace driver is busy (only one user access at the time is allowed).

```
An ODM system error has occurred
```

There can be two reasons for this message: either the user is not logged in as root or the ODM base has been corrupted and, therefore, the ODIT OPP needs to be re–installed.

## How to Change or Show MAD Trace Levels

### Access

From the ODIT – Change/Show OSI Layers Trace Levels, select:

```
Change/Show MAD Trace Levels
```

**FastPath**: `odit_st_mad`

**Command**: **ositrace –kc –mad –w** *Levels **

### Overview

The Change/Show MAD Trace Levels menu allows to display and modify the trace levels which are specific to the MAD (Distributed Access Method) kernel entity by giving access to the following fields:

DRIVER TRACE
MEMORY TRACE
TIMER TRACE
PERFORMANCE TRACE
FIFO LIST TRACE
ACCESS METHOD
INTERRUPT TRACE
MEMORY MANAGER TRACE
LOCK SYSTEM CALLS

### Description

This section describes the above listed fields.

Unless otherwise specified in the description of each individual field below, a trace level is enabled by setting the field to ON and disabled by setting it to OFF.

#### Driver Trace Field

This trace level provides information on all MAD actions and mechanisms.

#### Memory Trace Field

This trace level provides information on dynamic memory allocation by MAD modules.

#### Timer Trace Field

This trace level provides information concerning Timer module access.

#### Performance Trace Field

This trace level can be used to measure performance of OSI driver modules.

#### FIFO List Trace Field

This trace level provides information on ICB/UCB list management.

#### Access Method Field

This trace level provides information concerning MAD processing.

#### Interrupt Trace Field

This trace level provides information concerning all accesses to the OSI Driver Interrupt System.

#### Memory Manager Trace Field

This trace level provides information concerning memory manager processing.

### Lock System Calls Field

This trace level provides information concerning all accesses to the OSI Driver system locks and unlocks.

## Possible Errors

On selecting the function “Change/Show MAD Trace Levels”, the following messages may appear:

```
Permission denied! Super user only!
```

Super user access permission is required in order to use this function.

```
Warning! ODM configuration updated but OSI stack cannot be
accessed
```

When this message appears, either the OSI stack has not been loaded or the OSI trace driver is busy (only one user access at the time is allowed).

```
An ODM system error has occurred
```

There can be two reasons for this message: either the user is not logged in as root or the ODM base has been corrupted and, therefore, the ODIT OPP needs to be re–installed.

## How to Reset the Trace Buffer

### Access

From the ODIT – OSI Layers Trace Management menu, select:

```
Reset Trace Buffer
```

**FastPath**: `odit_reset`

**Command**: **ositrace –c –reset**

### Overview

The Reset Trace Buffer Menu is used to reset the OSI stack trace by resetting the trace buffer that contains the trace of all kernel entities managed via the OSI Layers Trace Management Menu, and the trace of Protocol Data Units (caught at MAC level) which are managed via the OSI Protocol Analyzer Menu.

### Possible Errors

On selecting the function “Reset Trace Buffer”, the following messages may appear:

```
Permission denied! Super user only!
```

Super user access permission is required in order to use this function.

```
Warning! ODM configuration updated but OSI stack cannot be
accessed
```

When this message appears, either the OSI stack has not been loaded or the OSI trace driver is busy (only one user access at the time is allowed).

```
An ODM system error has occurred
```

There can be two reasons for this message: either the user is not logged in as root or the ODM base has been corrupted and, therefore, the ODIT OPP needs to be re–installed.

## How to Perform Trace Analysis

### Access

From the ODIT – OSI Layers Trace Management menu, select:

```
Trace Analysis
```

**FastPath**: `odit_st_run`

**Command**: **ositrace –run –i** *mode* [**–fi** *filename*] **–o** *mode* [**–fo** *filename*]

### Overview

The Trace Analysis menu is used to run the MAD trace analysis and to generate a trace report. This trace report does not include API traces managed by AIX trace facilities. The Trace Analysis menu gives access to the following fields:

```
TRACE ENTRY MODE
FILENAME TO CREATE REPORT FROM
TRACE OUTPUT MODE
FILENAME TO WRITE REPORT TO (DEFAULT IS STDOUT)
```

### Description

This section describes the above listed fields. For a description of the trace report produced see “Trace Report” in Appendix A on page A-1.

#### Trace Entry Mode Field

This field allows to select the desired trace entry mode among the following available modes:

- Scan Mode: scans OSI stack trace messages provided by the MAD entity.
- Buffer Mode: analyzes OSI stack messages stored in the trace buffer.
- File Mode: analyzes OSI stack trace messages from a saved file.

The default mode is Buffer mode.

#### Filename to Create Report From Field

This field enables the user to specify the binary file that contains trace messages to analyze. It is meaningful only if File Mode has been previously selected (see above). Otherwise it is ignored.

There is no default value.

#### Trace Output Mode Field

This field enables the user to select one of the following display modes:

- Summary mode: provides a summary display of kernel entity trace information.
- Extended mode: provides a summary and a detailed display of kernel entity trace information.

The default display mode is Summary mode.

#### Filename to Write Report to (Default Is Stdout) Field

This optional field enables the user to specify the file to which are saved the results of the trace analysis.

The default is *stdout*.

## Possible Errors

On selecting the function “Trace Analysis”, the following messages may appear:

```
Permission denied! Super user only!
```

Super user access permission is required in order to use this function.

```
Warning! ODM configuration updated but OSI stack cannot be
accessed
```

When this message appears, either the OSI stack has not been loaded or the OSI trace driver is busy (only one user access at the time is allowed).

```
An ODM system error has occurred
```

There can be two reasons for this message: either the user is not logged in as root or the ODM base has been corrupted and, therefore, the ODIT OPP needs to be re–installed.

```
A system error has occurred. No such file or directory
```

The filename specified does not exist or cannot be created either because the same filename is already in use or because the pathname is incorrect.

## How to Save a Trace to a File

### Access

From the ODIT main menu, select:

```
OSI Trace Save Without Analysis
```

**FastPath**: `odit_save`

**Command**: **ositrace –save** *mode filename*

### Overview

This menu permits existing OSI stack traces to be saved to a file in binary format. This file can be used by the ODIT internal trace management or by the ODIT protocol analyzer to interpret OSI stack internal traces or to analyze Protocol Data Units.

**This function is reserved for Technical Support.**

This menu gives access to the following fields:

```
TRACE ENTRY MODE
FILENAME TO SAVE TRACE TO
```

### Description

This section describes the above listed fields.

#### Trace Entry Mode Field

- Scan Mode: scans the OSI stack trace buffer until the trace recording is stopped by the user with the break key.
- Buffer Mode: analyzes OSI stack messages stored in the trace buffer.
- File Mode: gets the OSI stack trace buffer.

The default mode is Buffer mode.

#### Filename to Save Trace To Field

This field enables the user to specify the binary file that will contain the current OSI stack trace. For instance:

```
File name               /var/tmp/osi/odit.tra
```

The file name of OSI Stack traces to be saved must be entered and confirmed with **Do**.

## Possible Errors

On selecting the function “OSI Trace Save Without Analysis”, the following messages may appear:

```
Permission denied! Super user only!
```

Super user access permission is required in order to use this function.

```
Warning! ODM configuration updated but OSI stack cannot be
accessed
```

When this message appears, either the OSI stack has not been loaded or the OSI trace driver is busy (only one user access at the time is allowed).

```
An ODM system error has occurred
```

There can be two reasons for this message: either the user is not logged in as root or the ODM base has been corrupted and, therefore, the ODIT OPP needs to be re–installed.

```
A system error has occurred. No such file or directory
```

The filename specified does not exist or cannot be created either because the same filename is already in use or because the pathname is incorrect.

# Chapter 8. OSI Protocol Analyzer

This chapter describes how to configure and run the ODIT protocol analyzer in the following sections:

- Accessing the OSI Protocol Analyzer on page 8-1,
- How to List Protocol Analyzer Configuration on page 8-3,
- How to Reset Protocol Analyzer Configuration on page 8-4,
- How to Tune the Protocol Analyzer on page 8-5,
- How to Set Analyzer Output Options on page 8-7,
- How to Set Analyzer Filter Options on page 8-9,
- How to Run the Protocol Analyzer on page 8-11.

It also explains how to analyze the trace of decoded Protocol Data Units caught at the level of the MAC and X25 layers. The following layers can be analyzed:

- MAC layer (Media Access Control), see page 8-13,
- LLC layer (Logical Link Control), see page 8-14,
- X25.3 layer (Network protocol for WAN), see page 8-15,
- ES–IS layer (End System, Intermediate System), see page 8-21,
- CLNP layer (ConnectionLess Network Protocol), see page 8-24,
- COTP layer (Connection Oriented Transport Protocol), see page 8-28,
- COSP layer (Connection Oriented Session Protocol), see page 8-33.

## Accessing the OSI Protocol Analyzer

### Access

From the ODIT main menu, select:

```
ODIT – OSI Protocol Analyzer
```

**FastPath**: `odit_prot`

### Overview

This menu gives access to the Protocol Analyzer operations.

```
ODIT –  OSI Protocol Analyzer

List Protocol Analyzer Configuration
Reset Protocol Analyzer Configuration
Protocol Analyzer Tuning
Analyzer Output Options
Analyzer Filter Options
Reset Trace Buffer
Protocol Analyzer Run
```

## Description

The OSI Protocol Analyzer menu enables the user to reset, tune, define options and run the protocol analyzer according to parameter selections previously entered. See the detailed information on the operations offered by this menu in the following sections.

List Protocol Analyzer Configuration
: Enables the user to list the protocol analyzer's configuration parameters. See "How to List Protocol Analyzer Configuration" on page 8-3.

Reset Protocol Analyzer Configuration
: Enables the user to reset the protocol analyzer's configuration parameters. See "How to Reset Protocol Analyzer Configuration" on page 8-4.

Protocol Analyzer Tuning
: Enables/disables the OSI stack protocol trace. See "How to Tune the Protocol Analyzer" on page 8-5.

Analyzer Output Options
: Enables the user to select the networks and layers to be displayed for the analysis output. See "How to Set Analyzer Output Options" on page 8-7.

Analyzer Filter Options
: Sets a filter on PDUs to be analyzed by selecting the MAC addresses and SAPs. See "How to Set Analyzer Filter Options" on page 8-9.

Reset Trace Buffer
: This facility is described in "How to Reset the Trace Buffer" on page 7-51.

Protocol Analyzer Run
: Enables the user to select the entry mode and output mode for protocol analysis and starts the analysis as determined by the configuration setup. See "How to Run the Protocol Analyzer" on page 8-11.

## How to List Protocol Analyzer Configuration

### Access

From the OSI Protocol Analyzer menu, select:

```
ODIT - List Protocol Analyzer Configuration
```

**FastPath**: `odit_pr_list`

**Command**: **osiprot –lc**

### Overview

This menu enables the user to display the current protocol analyzer configuration. This includes:

- selected input mode
- active filters
- active boards
- selected output mode
- selected OSI layers

### Description

This is an example of configuration display:

```
Input Mode          = BUFFER

No filters active

Output Mode         = SUMMARY
Medium Type         = X25+ETH+TKR+FDDI
OSI Layers Selected = MAC+X253+LLC+ESIS+CLNP+COTP+COSP

osiprot: Successful Completion
```

## How to Reset Protocol Analyzer Configuration

### Access

From the OSI Protocol Analyzer menu, select:

```
ODIT – Reset Protocol Analyzer Configuration
```

**FastPath**: `odit_pr_rconf`

**Command**: **osiprot –set –rc**

### Overview

This functions enables the user to reset the Protocol Analyzer configuration back to its initial default values.

### Description

The default values for the protocol configurable parameters are as follows:

```
Protocol trace = not active

PDU truncature = 128

Output = all layers all media

Filters = no filters

Input mode = buffer

Output mode = summary
```

### Possible Errors

On selecting the function “Reset Protocol Analyzer Configuration”, the following messages may appear:

```
Permission denied! Super user only!
```

Super user access permission is required in order to use this function.

```
Warning! ODM configuration updated but OSI stack cannot be
accessed
```

When this message appears, either the OSI stack has not been loaded or the OSI trace driver is busy (only one user access at the time is allowed).

```
An ODM system error has occurred
```

There can be two reasons for this message: either the user is not logged in as root or the ODM base has been corrupted and, therefore, the ODIT OPP needs to be re–installed.

## How to Tune the Protocol Analyzer

### Access

From the OSI Protocol Analyzer menu, select:

```
ODIT - Protocol Analyzer Tuning
```

**FastPath**: `odit_pr_tune`

**Command**: **osiprot –set –pt –w** *ProtTrace TruncatureSize*

### Overview

This menu function is used to enable or disable the Protocol trace and change the PDU truncature size by giving access to the following fields:

```
PROTOCOL TRACE
PDU TRUNCATURE SIZE (IN BYTES)
```

### Description

This section describes the above listed fields:

#### Protocol Trace Field

This trace level provides information on incoming and outgoing Protocol Data Units. When the screen appears the current value is retrieved from the OSI stack (kernel) and displayed.

To enable this trace level set field to ON, to disable, set to OFF.

The default value is the value previously configured. On a new system, the field is set to OFF.

#### PDU Truncature Size Field

This field enables the user to specify the maximum PDU length that will be considered by the Protocol Analyzer. The PDU length is a value that must be between 40 bytes and 1536 bytes.

The dialog field is as follows:

**PDU truncated size (in bytes)**
Enter size in bytes.
Limits are as follows:
40 < size < 1536.

The default value of the field is 128 bytes.

## Possible Errors

On selecting the function “Protocol Analyzer Tuning”, the following messages may appear:

```
Permission denied! Super user only!
```

Super user access permission is required in order to use this function.

```
The attribute must be within the range 40 –> 1536
```

In the two above messages, the value supplied is not within the specified range.

```
Warning! ODM configuration updated but OSI stack cannot be
accessed
```

When this message appears, either the OSI stack has not been loaded or the OSI trace driver is busy (only one user access at the time is allowed).

```
An ODM system error has occurred
```

There can be two reasons for this message: either the user is not logged in as root or the ODM base has been corrupted and, therefore, the ODIT OPP needs to be re–installed.

## How to Set Analyzer Output Options

### Access

From the OSI Protocol Analyzer menu, select:

```
ODIT – Analyzer Output Options
```

**FastPath**: `odit_pr_output`

**Command**: **osiprot –set –out –w** {**–med** *Medium*}* **–layer** {*Layers*}*

### Overview

This function enables the user to select the networks to be considered and which layers to display next time the Protocol Analyzer is run by giving access to the following fields:

```
MEDIUM TYPE
MAC LAYER
LLC LAYER
X25.3 LAYER
ES–IS LAYER
CLNP LAYER
COTP LAYER
COSP LAYER
```

### Description

This section describes the above listed fields.

Unless otherwise specified in the description of each individual field below, a trace level is enabled by setting the field to ON and disabled by setting it to OFF. The default value of a field is the value previously configured. On a new system, the fields are enabled (ON).

#### Medium Type Field

This field enables the user to select the communication adapters to be considered by the Protocol Analyzer. The list proposed is as follows:

- ETH = Ethernet 802.3
- TKR = Token Ring
- FDDI = Fibre Distributed Data Interface
- X25 = X.25 adapter board

The default values are those previously configured. On a new system, all values are selected.

#### MAC Layer Field

This field enables the user to select MAC layer for PDU analysis.

#### LLC Layer Field

This field enables the user to select LLC layer for PDU analysis.

#### X25.3 Layer Field

This field enables the user to select X25.3 layer for PDU analysis.

#### ES–IS Layer Field

This field enables the user to select ES–IS layer for PDU analysis.

#### CLNP Layer Field

This field enables the user to select CLNP layer for PDU analysis.

### COTP Layer Field

This field enables the user to select COTP layer for PDU analysis.

### COSP Layer Field

This field enables the user to select COSP layer for PDU analysis.

## Example

```
ODIT – Output Options

Medium type              ETH+TKR
MAC layer                OFF
LLC layer                OFF
X25.3 layer              OFF
ES–IS layer              OFF
CLNP layer               OFF
COTP layer               ON
CLTP layer               OFF
COSP layer               OFF
```

## Possible Errors

On selecting the function “Analyzer Output Options”, the following messages may appear:

```
Permission denied! Super user only!
```

Super user access permission is required in order to use this function.

```
Warning! ODM configuration updated but OSI stack cannot be
accessed
```

When this message appears, either the OSI stack has not been loaded or the OSI trace driver is busy (only one user access at the time is allowed).

```
An ODM system error has occurred
```

There can be two reasons for this message: either the user is not logged in as root or the ODM base has been corrupted and, therefore, the ODIT OPP needs to be re–installed.

## How to Set Analyzer Filter Options

### Access

From the OSI Protocol Analyzer menu, select:

```
ODIT – Analyzer Filter Options
```

**FastPath**: `odit_pr_filter`

**Command**: **osiprot –set –ft –w** *Filters*

### Overview

This function enables the user to filter the PDUs to be analyzed according to MAC addresses and Service Acess Points (SAPs). This function is managed locally by ODIT. The list of local MAC addresses and the list of the local NSAPs on the current station are given to the user.

**Note:** The current values managed locally by ODIT, are displayed for this screen. A field supplied with '*' means filter not active.

The following editable fields are available:

```
MAC – LOCAL ADDRESS
MAC – REMOTE ADDRESS
LLC – LOCAL LSAP
LLC – REMOTE LSAP
CLNP – LOCAL NSAP
CLNP – REMOTE NSAP
```

### Description

This section describes the above listed fields.

Unless otherwise specified in the description of each individual field, the default value of a field is the value previously configured. On a new system, this value is "*" (i.e. filter not active).

#### MAC – Local Address Field

This field enables the user to select local MAC addresses by entering a character string of 12 digits maximum in hexadecimal format.

#### MAC – Remote Address Field

This field enables the user to select remote MAC addresses by entering a character string of 12 digits maximum in hexadecimal format.

#### LLC – Local LSAP Field

This field enables the user to select local LSAPs by entering a character string of 2 digits maximum in hexadecimal format.

#### LLC – Remote LSAP Field

This field enables the user to select remote LSAPs by entering a character string of 2 digits maximum in hexadecimal format.

#### CLNP – Local NSAP Field

This field enables the user to select local NSAPs by entering a character string of 2 digits minimum and 40 maximum in hexadecimal format.

### CLNP – Remote NSAP Field

This field enables the user to select remote NSAPs by entering a character string of 2 digits minimum and 40 maximum in hexadecimal format.

The default value is the value previously configured. On a new system, this value is "*" (i.e. filter not active).

**Note:** A digit represents a hexadecimal character ('0' to 'F').
A '?' represents any hexadecimal character.
A '*' represents any hexadecimal string.
For example:
'0114' and '01A4' match the pattern '01?4'
'01234' and '01ABC' match the pattern '01*'

## Example

```
ODIT – Filter Options

MAC Local Address              02608C2C94B5
MAC Remote Address             02608C4D27B5
Link layer – Local LSAP        FE
Link layer – Remote LSAP       FE
Network layer – Local NSAP     0102*
Network layer – Remote NSAP    01020305
```

## Possible Errors

On selecting the function "Analyzer Filter Options", the following messages may appear:

```
Permission denied! Super user only!
```

Super user access permission is required in order to use this function.

```
Warning! ODM configuration updated but OSI stack cannot be
accessed
```

When this message appears, either the OSI stack has not been loaded or the OSI trace driver is busy (only one user access at the time is allowed).

```
An ODM system error has occurred
```

There can be two reasons for this message: either the user is not logged in as root or the ODM base has been corrupted and, therefore, the ODIT OPP needs to be re–installed.

## How to Run the Protocol Analyzer

### Access

From the OSI Protocol Analyzer menu, select:

```
Protocol Analyzer Run
```

**FastPath**: `odit_pr_run`

### Overview

This menu enables the user to run the Protocol Analyzer in the configuration defined in the “How to Tune the Protocol Analyzer” menu (see page 8-5) and to obtain an output corresponding to the display mode selected in “How to Set Analyzer Output Options” (see page 8-7). For detailed information on the output file produced, refer to “Trace Report” on page A-1.

This menu gives access to the following dialog fields:

```
PROTOCOL ENTRY MODE
FILENAME TO CREATE REPORT FROM
OUTPUT MODE
FILENAME TO WRITE REPORT TO (DEFAULT IS STDOUT)
```

### Description

This section describes the above listed fields.

#### Protocol Entry Mode Field

This field enables the user to select the desired trace entry mode among the three modes available:

- Scan mode: scans current OSI stack trace messages provided by the MAD.
- Buffer mode: analyzes OSI stack trace messages stored in the trace buffer.
- File mode: analyzes OSI stack trace messages from a saved file.

The dialog field is as follows:

| | |
|---|---|
| **Input Mode** | Choose from list of entry modes:<br>**Scan**<br>**Buffer**<br>**File**.<br>Where:<br>Scan = Scans current PDUs provided by OSI driver.<br>Buffer = Analyzes PDUs stored in the OSI Stack internal buffer.<br>File = Analyzes PDUs saved in a file. |

**Note:** There is no default value displayed for this screen.The default value of the Input Mode field is the mode previously configured. On a new system, Buffer mode is configured.

This dialog is used to enter the name of a previously saved OSI stack trace file. The file is generated under the “OSI Trace Save Without Analysis” menu, see “How to Save a Trace to a File” on page 7-54.

#### Filename To Create Report From Field

This optional field enables the user to specify the binary file that contains the protocol trace to analyze.

The default value is the value previously configured. On a new system, the default value is */var/tmp/osi/odit.tra*.

### Output Mode Field

This field enables the user to select the desired display mode among the three modes available:

- Summary mode: provides a summary display of PDU headers (main information).
- Hexadecimal mode: provides a summary display and a hexadecimal dump of PDU headers.
- Comprehensive mode: provides a summary and a detailed display of PDU headers.

### Filename to Send Report to (Default is Stdout) Field

This field enables the user to specify the file that will contain the result of the protocol trace analysis.

The default value for this field is stdout.

## Possible Errors

On selecting the function “Protocol Analyzer Run”, the following messages may appear:

```
Permission denied! Super user only!
```

Super user access permission is required in order to use this function.

```
A system error has occurred. No such file or directory
```

The filename specified does not exist or cannot be created either because the same filename is already in use or because the pathname is incorrect.

## How to Analyze MAC Layer Traces

### Access

From the Analyzer Output Options menu, select the field:

```
MAC layer
```

The following trace summary is displayed:

**MAC Layer**

| Time | No | Size | Way | Type | LocMACaddr | RemMACaddr |
|---|---|---|---|---|---|---|
| 10.06.42.71 0 | 001 | 30 | ⇒ | Ethernet | 0x02608c2c8b79 | 0x9002b000004 |
| 10.06.52.290 | 002 | 74 | ⇒ | Ethernet | 0x02608c2c8b79 | 0x02608c2fa32b |
| 10.06.52.850 | 003 | 70 | ⇐ |  | 0x02608c2fa32b | 0x02608c2c8b79 |

### Field descriptions

| | |
|---|---|
| Time | The time when the PDU was transmitted to (or from) adjacent layer. |
| No | Running count of PDUs. |
| Size | PDU size in bytes. Decimal format, 5 digits. |
| Way | Incoming or Outgoing PDUs.<br>The display format for this field is:<br>⇒ PDU sent to remote station.<br>⇐ PDU received from remote station. |
| Type | Medium type<br>Possible values are:<br>Ethernet<br>Token Bus<br>Token Ring<br>FDDI |
| LocMACaddr | Local MAC address. Hexadecimal format, 12 digits. |
| RemMACaddr | Remote MAC address. Hexadecimal format, 12 digits. |

# How to Analyze LLC Layer Traces

## Access

From the Analyzer Output Options menu, select the field:

```
LLC layer
```

The following trace summary is displayed:

**LLC Layer**

| Time | No | Size | Way | Type | Errs | Lsap | Rsap | Frame Type |
|---|---|---|---|---|---|---|---|---|
| 18:03.55.030 | 001 | 70 | ⇐ | UI | | fe | fe | Information |
| 18:03.57.130 | 002 | 70 | ⇒ | UI | | fe | fe | Information |
| 18:04.00.240 | 003 | 70 | ⇐ | TEST | | fe | fe | TEST Command |
| 18:04.10.120 | 004 | 6 | ⇒ | XID | | 0 | 0 | XID Command |
| 18:04.15.450 | 005 | 6 | ⇒ | XID | LENG? | fe | fe | XID Command |
| 18.04.34.510 | 006 | 15 | ⇒ | ?6b | PTYP? | | | ?Unknown Cmd |

## Field descriptions

| | |
|---|---|
| Time | Time when LLC PDU was transmitted to (or from) adjacent layer. |
| No | Running count of LLC PDU's. |
| Size | LLC PDU size in bytes. Decimal format, 5 digits. |
| Way | Incoming or Outgoing PDUs.<br>The display format for this field is:<br>⇒ LLC PDU sent to remote station.<br>⇐ LLC PDU received from remote station. |
| Type | LLC PDU's are listed below: |

| | |
|---|---|
| TEST | Test |
| UI | Un–numbered information |
| XID | Exchange identification |
| ?<code> | PDU type unknown |

Errs Error message field with the following possible values:

| | |
|---|---|
| LENG? | LENGth too long for a valid frame. |
| PTYP? | Pdu TYPe is invalid, must be:<br>TEST(f3), UI(03) or XID(bf). |

Lsap Address of Local LSAP displayed in hexadecimal format.
Local LLC Service Access Point

Rsap Address of Remote LSAP displayed in hexadecimal format.
Remote LLC Service Access Point

Frame Type Type of LLC PDU.
The possible LLC PDUs are:
Information
TEST Command
XID Command
?Unknown Cmd.

## How to Analyze X25.3 Layer Traces

### Access

From the Analyzer Output Options menu, select the field:

```
X25.3 layer
```

The following trace summary is displayed:

**X25.3 Layer**

| Time | No | Size | Way | Type | Errs | Icn | Lcn | DQM | Data | Parameters |
|---|---|---|---|---|---|---|---|---|---|---|
| 13:21.15.030 | 001 | 44 | ⇒ | CAr | | 0x4 | 65 | Y | 12 | xyf(PsWs) |
| 13:21.24.070 | 002 | 47 | ⇐ | CAc | | 0x4 | 65 | Y | 15 | xyf(PsWs) |
| 13:22.03.340 | 003 | 66 | ⇐ | DT | | 0x4 | 65 | YNN | 63 | |
| 13:22.27.450 | 004 | 123 | ⇒ | DT | | 0x4 | 65 | NNN | 120 | |
| 13:22.35.120 | 005 | 3 | ⇐ | RR | | 0x4 | 65 | | | |
| 13:22.38.340 | 006 | 5 | ⇒ | CLr | | 0x4 | 65 | | | c(0)d(49) |
| 13:22.45.660 | 007 | 3 | ⇐ | CLc | | 0x4 | 65 | | | |
| 13.22.54.130 | 008 | 34 | ⇒ | ?fe | PTYP? | | | | | |

### Field descriptions

| | |
|---|---|
| Time | Time when X25.3 packet was transmitted to (or from) adjacent layer. |
| No | Running count of X25.3 packets. |
| Size | X25.3 packet size in bytes. Decimal format, 4 digits. |
| Way | Incoming or Outgoing X25.3 packet.<br>The display format for this field is:<br>⇒ X25.3 packet sent.<br>⇐ X25.3 packet received. |

Type — X25.3 packet types are listed below:

| | |
|---|---|
| CAc | Call Confirmation |
| CAr | Call Request (or Incoming Call) |
| CLc | Clear Confirmation |
| CLr | Clear Request (or Clear Indication) |
| DT | Data |
| ITc | Interrupt Confirmation |
| ITr | Interrupt Request (or Interrupt Indication) |
| REc | Reset Confirmation |
| REJ | Reject |
| REr | Reset Request (or Reset Indication) |
| RNR | Receive Not Ready |
| RR | Receive Ready |

Errs — Error message field. The possible values are:

| | |
|---|---|
| ADIN? | DTE ADdress is INvalid |
| BLCE? | Bad Length for CallEd DTE address |
| BLCI? | Bad Length for CallIng DTE address |
| CAIN? | CAuse field is INvalid |
| CNIN? | Channel Number is INvalid |
| FLIN? | Facility Length is INvalid |
| FPCI? | Facility Parameter Code is Invalid |
| FPIP? | Facility Parameter is Invalid for this X25.3 Packet |
| FPVI? | Facility Parameter Value is Invalid |
| LAIC? | Length of DTE Addresses Inconsistent with Contents |
| LFIC? | Length of Facilities Inconsistent with Contents |

| | | |
|---|---|---|
| | PTYP? | Pdu TYPe is invalid |

| | |
|---|---|
| Icn | Internal channel number. Hexadecimal format, 8 digits maximum. The display format is 0x<hexadecimal value>. |
| Lcn | Logical channel number. Decimal format, 4 digits. Must be between 1 and 4096. |
| DQM | D, Q, M flags concerning X25.3 data packets (DT) are: |

| | |
|---|---|
| D | Acknowledgements through to through required. |
| Q | Qualified data. |
| M | More data. |

A ”Y” indicates the flag is on, and a ”N” indicates the flag is off.

**NOTE**: The D flag is also meaningful for the following X25.3 packet types:

| | |
|---|---|
| CAc | Call Confirmation |
| CAr | Call Request (or Incoming Call). |

Data — Number of user data carried in the X25.3 packet in bytes. Decimal format, 4 digits.

Meaningful only for the following X25.3 packet types:

| | |
|---|---|
| CAc | Call Confirmation |
| CAr | Call Request (or Incoming Call) |
| CLr | Clear Request (or Clear Indication) |
| ITr | Interrupt Request (or Interrupt Indication) |
| DT | Data |

Parameters — Displays other X25.3 packet parameters depending on the packet type. If a parameter is present, a single letter is listed for it.
The following parameters are available for X25.3 packets:

Cause–diagnostic parameters (meaningful for CLr and REr packets)

| | |
|---|---|
| c | Cause. Cause value is displayed between ”( )”. |
| d | Diagnostic. The diagnostic value is displayed between ”( )”. |

DTE addresses (meaningful for CAr, CAc, CLr and CLc packets)

| | |
|---|---|
| x | Calling DTE address present. |
| y | Called DTE address present. |

Facilities (meaningful for CAr, CAc, CLr and CLc packets)

| | |
|---|---|
| f | Facilities. Facilities are displayed between ”[]”. For each facility, a mnemonic code is listed for it. |

User Data (meaningful for CAr, CAc, CLr, DT, ITr packets)

| | |
|---|---|
| u | User Data. No value is displayed. |

for the ”c” parameter:
The display format is as follows:

c(x)

Possible Cause values (for CLr packet type) are:

| | |
|---|---|
| 0 or 128 | DTE Originated |
| 1 or 129 | Number Busy |
| 3 or 131 | Invalid Facility Request |
| 5 or 133 | Network Congestion |
| 9 or 137 | Out of Order |
| 11 or 139 | Access Barred |
| 13 or 141 | Not Obtainable |
| 17 or 145 | Remote Procedure Error |
| 19 or 147 | Local Procedure Error |
| 21 or 149 | RPOA Out of Order |
| 25 or 153 | Reverse Charging Acceptance Not Subscribed |
| 33 or 161 | Incompatible Destination |

| | |
|---|---|
| 41 or 169 | Fast Select Acceptance Not Subscribed |
| 57 or 185 | Ship Absent |
| 193 | Gateway–detected Procedure Error |
| 195 | Gateway Congestion |
| <code>? | Cause unknown. |

Possible Cause values (for REr packet type) are:

| | |
|---|---|
| 0 or 128 | DTE Originated |
| 1 or 129 | Out of Order |
| 3 or 131 | Remote Procedure Error |
| 5 of 133 | Local Procedure Error |
| 7 or 135 | Network Congestion |
| 9 or 137 | Remote DTE Operational |
| 15 or 143 | Network Operational |
| 33 or 161 | Incompatible Destination |
| 45 or 173 | Network Out of Order |
| 193 | Gateway–detected Procedure Error |
| 195 | Gateway Congestion |
| 199 | Gateway Operational |
| <code>? | Cause unknown. |

For the ”d” parameter, the display format is as follows:

d(x).

Diagnostic code possible values are:

| | |
|---|---|
| **–15** | **No Additional Information** |
| 1 | Invalid P(S) |
| 2 | Invalid P(R) |
| **16–31** | **Packet Type Invalid** |
| **32–47** | **Packet Not Allowed** |
| 33 | Unidentifiable Packet |
| 34 | Call on one–way logical channel |
| 35 | Invalid packet type on a Permanent Virtual Circuit |
| 36 | Packet on an unassigned logical channel |
| 37 | Reject not subscribed |
| 38 | Reject not subscribed to packet too short |
| 39 | Reject not subscribed to packet too long |
| 40 | Invalid General Format Identifier |
| 41 | Restart of Registration packet with nonzero Logical Channel Identifier |
| 42 | Packet type not compatible with facility |
| 43 | Unauthorized Interrupt Confirmation |
| 44 | Unauthorized Interrupt |
| 45 | Unauthorized Reject |
| **48–63** | **Timer Expired** |
| **64–79** | **Call Setup, Call Clearing, or Registration Problem** |
| 65 | Facility/Registration code not allowed |
| 66 | Facility parameter not allowed |
| 67 | Invalid Called DTE Address |
| 68 | Invalid Calling DTE Address |
| 69 | Invalid Facility/Registration length |
| 70 | Incoming call barred |
| 71 | No Logical Channel available |
| 72 | Call Collision |
| 73 | Duplicate Facility requested |
| 74 | Nonzero Address length |
| 75 | Nonzero Facility length |
| 76 | Facility not provided when expected |
| 77 | Invalid CCITT–specified DTE facility |
| 78 | Maximum number of call redirections or call deflections exceeded |

| | |
|---|---|
| **80–95** | **Miscellaneous** |
| 81 | Improper cause code from DTE |
| 82 | Nonoctet aligned |
| 83 | Inconsistent Q–bit settings |
| 84 | NUI problem |
| **96–111** | **Not Assigned** |
| **112–127** | **International Problem** |
| 113 | Remote network problem |
| 114 | International protocol problem |
| 115 | International link out of order |
| 116 | International link busy |
| 117 | Transit network facility problem |
| 118 | Remove network facility problem |
| 119 | International routing problem |
| 120 | Temporary routing problem |
| 121 | Unknown called DNIC |
| 122 | Maintenance action |
| **128–143** | **Reserved for DTE–defined Diagnostic Information** |
| **144–159** | **Timer Expired or Retransmission Count Surpassed** |
| **160–175** | **DTE–Specific Signals** |
| 161 | DTE operational |
| 162 | DTE not operational |
| 163 | DTE resource constraint |
| 164 | Fast Select not subscribed |
| 165 | Invalid partially full Data packet |
| 166 | D–bit procedure not supported |
| 167 | Registration/Cancellation confirmed |
| **176–223** | **Not Assigned** |
| **224–239** | **OSI Network Service Problem** |
| 225 | Disconnection (transient condition) |
| 226 | Disconnection (permanent condition) |
| 227 | Connection rejection – reason unspecified (transient condition) |
| 228 | Connection rejection – reason unspecified (permanent condition) |
| 229 | Connection rejection – quality of service not available (transient condition) |
| 230 | Connection rejection – quality of service not available (permanent condition) |
| 231 | Connection rejection – NSAP unreachable (transient condition) |
| 232 | Connection rejection – NSAP unreachable (permanent condition) |
| 233 | Reset – reason unspecified |
| 234 | Reset – congestion |
| 235 | Connection rejection – NSAP address unknown (permanent condition) |
| **240–255** | **Higher Layer Initiated** |
| 241 | Disconnection – normal |
| 242 | Disconnection – abnormal |
| 243 | Disconnection – incompatible information in user data |
| 244 | Connection rejection – reason unspecified (transient condition) |
| 245 | Connection rejection – reason unspecified (permanent condition) |
| 246 | Connection rejection – quality of service not available (transient condition) |
| 247 | Connection rejection – quality of service not available (permanent condition) |

| | |
|---|---|
| 248 | Connection rejection – incompatible information in user data |
| 249 | Connection rejection – unrecognizable protocol identifier in user data |
| 250 | Reset – user resynchronisation |

For the ”f” parameter, the display format is as follows:

f(F1F2. . .Fn)

where Fi is an X25.3 facility mnemonic listed below:

| | |
|---|---|
| **Ps** | Packet size negotiation |
| **Ws** | Window size negotiation |
| **Tc** | Throughput class negotiation |
| **G1** | Basic Closed User Group Selection |
| **G2** | Extended Closed User Group Selection |
| **G3** | Basic Closed User Group with outgoing access selection |
| **G4** | Extended Closed User Group with outgoing access selection |
| **G5** | Bilateral Closed User Group selection |
| **Fr** | Fast select and reverse charging |
| **Ns** | Nui selection |
| **Rs** | Requesting service (charging information) |
| **Im** | Indicating monetary unit (charging information) |
| **Is** | Indicating segment count (charging information) |
| **Ic** | Indicating call duration (charging information) |
| **Ic** | Indicating call duration (charging information) |
| **Rb** | RPOA selection (basic) |
| **Re** | RPOA selection (extended) |
| **Cd** | Call deflection selection |
| **Cr** | Call redirection or detection notification |
| **Cl** | Called line address modified notification |
| **Td** | Transit delay selection and indication |
| **Mk** | Marker transpac |
| **Ci** | Calling address extension |
| **Ce** | Called address extension |
| **Mt** | Minimum throughput class negotiation |
| **Ed** | End–to–End transit delay negotiation |
| **Py** | Priority |
| **Pt** | Protection |
| **Ex** | Expedited data negotiation |

(nn?) is listed in place of the mnemonic if the facility code is unknown where nn is the hexadecimal value of the facility code.
! is listed immediately after the mnemonic if the facility length or the facility value is incorrect.

**X25.3 Layer – Extended Display**

ODIT presents an extended display of X25.3 packets which completes the summary display described above. The following list gives all information that may appear under summary line, when extended display mode is selected.

For X25.3, the packet type description is displayed.

For example:

| | |
|---|---|
| (CAr) | Call Request |
| (ITc) | Interrupt Confirmation |

**For DTE addresses:**

| | |
|---|---|
| (x) | Calling DTE address = <value> |
| (y) | Called DTE address = <value> |

For example:

| | |
|---|---|
| (x) | Calling DTE address = 138010201 |
| (y) | Called DTE address = 138010202 |

**For cause–diagnostic fields:**

| | |
|---|---|
| (c) | Cause = <Cause text> |
| (d) | Diagnostic = <Diagnostic text> |

For example:

| | |
|---|---|
| (c) | Cause = DTE originated |
| (d) | Diagnostic = DTE not operational |

**For facilities:**

| | |
|---|---|
| (f) | Facilities (<facility mnemonic>) <facility text> |

**NOTE**: Each facility value is decoded and displayed in a symbolic format under the according facility line described above.

For example:

| | |
|---|---|
| (f) | Facilities |
| (Ps) | Packet size negotiation<br>From DTE = 128 bytes<br>To DTE = 128 bytes |

**For Data:**

| | |
|---|---|
| (u) | Call User Data |
| (u) | Called User Data |
| (u) | Clear User Data |
| (u) | Interrupt User Data |
| (u) | User Data |

Data is displayed in hexadecimal format.

## How to Analyze ES–IS Layer Traces

### Access

From the Analyzer Output Options menu, select the field:

```
ES–IS layer
```

The following trace summary is displayed:

**ES–IS Layer**

| **Time** | **No** | **Size** | **Way** | **Type** | **Errs** | **HoT** | **Opt** |
|---|---|---|---|---|---|---|---|
| 13:21.15.010 | 001 | 70 | ⇐ | ESH | | 40 | |
| 13:21.24.050 | 002 | 70 | ⇒ | ISH | | 20 | c |
| 13:22.03.240 | 003 | 70 | ⇐ | RD | | 30 | |
| 13:22.27.370 | 004 | 70 | ⇐ | ?1b | PTYP ? | | |

### Field descriptions

| | |
|---|---|
| Time | Time when NPDU was transmitted to (or from) adjacent layer. |
| No | Running count of NPDUs. |
| Size | NPDU size in bytes. Decimal format, 5 digits. |
| Way | Incoming or Outgoing PDUs, with the following display format: |
| | ⇒ ES–IS PDU sent to remote station. |
| | ⇐ ES–IS PDU received from remote station. |
| Type | ES–IS PDUs are listed below: |

| | |
|---|---|
| ESH | End system hello |
| ISH | Intermediate system hello |
| RD | Redirect |
| ?<code> | PDU type unknown. |

Errs — Error message field. The possible values are:

| | |
|---|---|
| AMTL? | Address Mask is To Long |
| IPRV? | Invalid PRiority Value |
| IQSP? | Invalid Quality of Service Parameter |
| ISEV? | Invalid SEcurity Value |
| LECB? | Length of End system Configuration time is Bad |
| LFFR? | Length of 255 (FF) is Reserved and may not be used |
| LHSF? | Length of indicator of Header too Short for Fixed part |
| LIIP? | Length of Indicator Inconsistent with Pdu contents |
| NES1? | There must be at least 1 Nsap in ESh pdu's |
| PIDI? | Protocol IDentifier Invalid, not equal to 0 or 0x81 or 0x82 |
| PIPD? | Parameter is Invalid for this PDu |
| PPN1? | Priority Parameter is Not 1 octet long |
| PTYP? | Pdu TYPe is invalid, must be: 2 (ESH), 4 (ISH), or 6 (RD) |
| RFZE | Reserved Field must be ZEro |
| SNML? | SNpa Mask is too Long for snpa |
| UPTO? | Unknown Parameter Type in Options part |
| VER1? | VERsion 1 must be used |

HoT — Hold time in one–second intervals.

| | |
|---|---|
| Opt | The variable parameters in the NPDU. If a parameter is present, a single letter is listed for it. |

(nn?) is listed in place of the letter if the parameter code is unknown, where nn is the hexadecimal value of the parameter code.

! is listed immediately after the letter if the parameter value is incor-rect.For example:

```
c(47?)p!
```

Where 0x47 is an unknown code parameter and priority value is invalid.

The following options are available for ES–IS PDU's:

| | |
|---|---|
| a | Address mask in RD PDU (Redirect PDU only) |
| c | Checksum |
| e | Redirect is to an end system (RD PDU only) |
| i | Redirect is to an intermediate system (RD PDU only) |
| m | SNPA mask in RD PDU (RD PDU only) |
| n | Number of NSAPs (ESH PDU only) |
| p | Priority |
| q | Quality of service |
| s | Security |
| t | Suggested end system configuration time (ISH PDU only) |

For the "n" option, the number of NSAPs is displayed as follows:

n(dd) where dd is a decimal value (nb NSAP's in ESH PDU)

**ES–IS – Extended Display**

ODIT presents a extended display of PDUs (ES–IS) which completes the summary display described above.
The following list gives all information that may appear under summary line, when extended display mode is selected. For the fixed part of PDU, the ESIS PDU type description is displayed.

For example:

| | |
|---|---|
| (ESH) | End System Hello |
| (c) | Checksum value = <value> |

**For addressing part of PDU**

| | |
|---|---|
| (e) | Redirect is to an end system |
| (i) | Redirect is to an intermediate system |
| (n) | Source Address (NSAP)<br>No <no> = <value><br>Network Entity Title<br>(NET) = <value><br>Destination Address<br>(DA) = <value><br>Subnetwork Address<br>(SNPA) = <value> |

**For option part of PDU**

|  |  |
|---|---|
| (a) | Address Mask value = <value> |
| (m) | SNPA Mask value = <value> |
| (p) | Priority value = <value> |
| (q) | Quality of service type = <type>, value = <value> |
| (s) | Security type = <type>, value = <value> |
| (t) | Suggested ES Configuration Time value = <value> |

**For errors on PDU**

|  |  |
|---|---|
| ??? | Error message = <Text of error> |

## How to Analyze CLNP Layer Traces

### Access

From the Analyzer Output Options menu, select the field:

```
CLNP layer
```

The following trace summary is displayed:

**CLNP Layer**

| **Time** | **No** | **Size** | **Way** | **Type** | **Errs** | **LiT** | **SME** | **DUI** | **SgOf** | **Data** | **Opt** |
|---|---|---|---|---|---|---|---|---|---|---|---|
| 13:21.15.030 | 001 | 70 | ⇐ | DT | | 40 | YNY | 1232 | 0 | 28 | cp |
| 13:21.24.070 | 002 | 70 | ⇒ | ERR | | 20 | YNY | 1234 | 0 | | cp |
| 13:22.03.340 | 003 | 70 | ⇐ | NUL | | | | | | | |
| 13:22.27.450 | 004 | 70 | ⇐ | ?1b | PTYP ? | | | | | | |

### Field descriptions

| | |
|---|---|
| Time | The time when the NPDU was transmitted to (or from) adjacent layer. |
| No | A running count of NPDU's. |
| Size | NPDU size in bytes. Decimal format, 5 digits. |
| Way | Incoming or Outgoing PDUs with the following display format: |

| | |
|---|---|
| ⇒ | CLNP PDU sent to remote station. |
| ⇐ | CLNP PDU received from remote station. |

Type

Network NPDU's are listed below:

| | |
|---|---|
| DT | data |
| ERR | Error |
| NUL | Null layer |
| ?<code> | PDU type unknown. |

| | |
|---|---|
| Errs | Error message field with possible values: |

| | |
|---|---|
| ERON? | Error Reporting flag ON for Error PDU |
| IPRV? | Invalid PRiority Value |
| IQSP? | Invalid Quality of Service Parameter |
| IRRT? | Invalid Record Route Termination status |
| ISEV? | Invalid SEcurity Value |
| ISRT? | Invalid Source Routing parameter Type |
| LFFR? | Length of 255 (FF) is Reserved and may not be used |
| LHSF? | Length of indicator of Header too Short for Fixed part |
| LIIP? | Length of Indicator Inconsistent with Pdu contents |
| LRDI? | Length of Reason for Discard parameter Invalid |
| LTZE? | LifeTime equal to ZEro |
| MSON? | More Segments flag ON for error pdu |
| NSLI? | NSap length inconsistent with LI |
| PIDI? | Protocol IDentifier Invalid, not equal to 0 or 0x81 or 0x82 |
| PIPD? | Parameter is Invalid for this PDu |
| PPN1? | Priority Parameter is Not 1 octet long |
| PTYP? | Pdu TYPe is invalid, must be DT(1c) or ERR(01) |
| SLGT? | Segment Length Greater than Total length |
| SLUP? | Segment Length Unequal to Pdu size |
| SNMT? | Segmentation Not permitted, More flag is True |
| SPON? | Segmentation Permitted flag ON for error pdu |
| UPTO? | Unknown Parameter Type in Options part |
| VER1? | VERsion 1 must be used |

| | |
|---|---|
| LiT | PDU LifeTime in half–second intervals. |
| SME | Header flags are: |

| | |
|---|---|
| S | Segmentation Permitted |
| M | More Segments |
| E | Error Report |

A ”Y” indicates the flag is on, and a ”N” indicates the flag is off.

| | |
|---|---|
| DUI | Data Unit Identifier, a unique identifier number for each initial PDU. |
| SgOf | Segment Offset, a position in the initial NPDU where the data belongs. |
| Data | Amount of data carried in the NPDU in bytes. A number, in decimal format, 5 digits. |

| | |
|---|---|
| Opt | The variable parameters in the NPDU. If a parameter is present, a single letter is listed for it. |

(nn?) is listed in place of the letter if the parameter code is unknown, where nn is the hexadecimal value of the parameter code.

! is listed immediately after the letter if the parameter value is incorrect.

For example:

```
c(47?)p!
```

Where 0x47 is an unknown code parameter and priority value is invalid.

The following options are available for CLNP PDU’s:

| | |
|---|---|
| c | Checksum |
| d | Reason for Discard |
| g | Padding |
| p | Priority |
| q | Quality of service |
| r | Recording of route |
| s | Security |
| u | Source routing |

For the ”d” option:

This option consists of two decimals values. The first one is **Reason** and the second one is **Offset** with the display format as follows:

```
d(x|x)
```

Values for reason are:

| | |
|---|---|
| 0 | Reason not specified |
| 1 | Protocol procedure error |
| 2 | Incorrect checksum |
| 3 | PDU discarded due to congestion |
| 4 | Header syntax error (cannot be parsed) |
| 5 | Segmentation needed but not permitted |
| 6 | Incomplete PDU received |
| 7 | Duplicate option |
| 128 | Destination address unreachable |
| 129 | Destination address unknown |
| 144 | Unspecified source routing error |
| 145 | Syntax error in source routing field |
| 146 | Unknown address in source routing field |
| 147 | Path not acceptable |
| 160 | LifeTime expired while data unit in transit |
| 161 | LifeTime expired during reassemble |
| 176 | Unsupported option not specified |
| 177 | Unsupported protocol version |
| 178 | Unsupported security option |
| 179 | Unsupported source routing option |
| 180 | Unsupported recording of route option |
| 192 | Reassemble interference |
| <code> | Unknown reason |

Offset value:

Error localised to a particular field or zero.

**CLNP Layer – Extended Display**

ODIT presents a extended display of PDUs (CLNP) which completes the summary display described above.

The following list gives all information that may appear under summary line, when extended display mode is selected.

For fixed part of PDU:

The NPDU type description is displayed.

For example:

| | |
|---|---|
| (DT) | Data NPDU |
| (c) | Checksum value = <value> |

For addressing part of PDU:

Local Address (NSAP) = <value>
Remote Address (NSAP) = <value>

For option part of PDU:

| | |
|---|---|
| (d) | Reason for Discard = <Reason text> |
| (g) | Padding value = <value> |
| (p) | Priority value = <value> |
| (q) | Quality of Service type = <type> , value = <value> |
| (r) | Recording of Route type = <type>, value = <value> |
| (s) | Security type = <type>,   value = <value> |
| (u) | Source Routeing type = <type>, value = <value> |

For errors on PDU:

| | |
|---|---|
| ??? | Error message = <Text of error> |

## How to Analyze COTP Layer Traces

### Access

From the Analyzer Output Options menu, select the field:

```
COTP layer
```

The following trace summary is displayed:

**COTP Layer**

| **Time** | **No** | **Size** | **Way** | **Type** | **Errs** | **Loc** | **Rem** | **NR** | **EOT** | **C** | **Data** | **C O** | **ECR** | **Opt** |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| 16:27:29:060 | 001 | 70 | ⇒ | CR | | 0e90 | 0000 | | | 8 | 0 | 4N | EC | cz11xyo |
| 16:27:37:050 | 002 | 70 | ⇐ | CC | | 0e90 | b789 | | | 8 | 0 | 4N | EC | z11xyo |
| 16:28:21:560 | 003 | 70 | ⇒ | DT | | 0e90 | b789 | 0 | T | | 62 | | | cp |
| 16:28:38:340 | 004 | 70 | ⇐ | AK | | 0e90 | b789 | 1 | | 8 | | | | |
| 16:29:11:130 | 005 | 70 | ⇒ | DR | | 0e90 | b789 | | | | | | | |
| 16:30:32:450 | 006 | 70 | ⇐ | ?1b | PTYP? | | | | | | | | | |

### Field descriptions

| | |
|---|---|
| Time | The time when the TPDU was transmitted to (or from) adjacent layer. |
| No | A running count of TPDU's. |
| Size | TPDU size in bytes. Decimal format, 5 digits. |
| Way | Incoming or Outgoing PDUs, with display format: |

| | |
|---|---|
| ⇒ | COTP PDU sent to remote station. |
| ⇐ | COTP PDU received from remote station. |

Type — COTP TPDU's are listed below:

| | |
|---|---|
| AK | Acknowledgement |
| CC | Connection Confirm |
| CR | Connection Request |
| DC | Disconnection Confirm |
| DR | Disconnection Request |
| DT | Data |
| EA | Expedited Acknowledgement |
| ED | Expedited Data |
| ER | Error |
| RJ | Reject |
| ?<code> | PDU type unknown |

The following CLTP TPDU is also considered:

| | |
|---|---|
| UD | Unit of Data |

Errs — Error message field with possible values:

| | |
|---|---|
| AON1? | Additional Options parameter is Not 1 octet long |
| CKC4? | ChecKsum shall be used only in Class 4 |
| DRZE? | Destination Reference number must be ZEro |
| ICLV? | Invalid CLass Value |
| ITPV? | Invalid TPdu size Value |
| LAKI? | Length of AcKnowledgement time parameter Invalid |
| LCKI? | Length of ChecKsum parameter Invalid |
| LFCI? | Length of Flow Control confirmation Invalid |
| LFFR? | Length of 255 (FF) is Reserved and may not be used |
| LHSF? | Length of indicator of Header too Short for Fixed part |
| LIIP? | Length of Indicator Inconsistent with Pdu contents |
| LITI? | Length of Incativity Timer Invalid |
| LPPI? | Length of Priority Parameter Invalid |
| LRRI? | Length of Residual error Rate Invalid |
| LRTI? | Length of Reassignment Time Invalid |
| LSNI? | Length of Subsequence Number Invalid |
| LTDI? | Length of Transit Delay parameter Invalid |
| LTPI? | Length of ThrouhPut parameter Invalid |
| PIPD? | Parameter is Invalid for this PDu |
| PTYP? | Pdu TYPe is invalid, must be: 2 (ESH), 4 (ISH), or 6 (RD) |
| TPN1? | Tpdu size Parameter is Not 1 octet long |
| UPTO? | Unknown Parameter Type in Options part |
| VER1? | VERsion 1 must be used |
| VPN1? | Version Parameter is Not 1 octet long |

Loc — Local reference number printed in hexadecimal format.

Rem — Remote reference number printed in hexadecimal format.

NR — Sequence number concerning only the following TPDU's:

| | |
|---|---|
| AK | Acknowledgement |
| DT | Data |
| EA | Expedited Acknowledgement |
| ED | Expedited Data |
| RJ | Reject |

EOT — End of TSDU marker concerning only the following TPDU's :

| | |
|---|---|
| DT | Data |
| ED | Expedited Data |

Possible values are:

| | |
|---|---|
| F | False |
| T | True |

C — Credit concerning only the following TPDU's:

| | |
|---|---|
| AK | Acknowledgement |
| CC | Connection Confirm |
| CR | Connection Request |
| RJ | Reject |

Data — TPDU user data size in bytes. Decimal format, 5 digits. Meaningful only for the following TPDU's:

| | |
|---|---|
| CC | Connection Confirm |
| CR | Connection Request |
| DR | Disconnection Request |
| DT | Data |
| ED | Expedited Data |
| UD | Unit of Datas |

| | |
|---|---|
| CO | Class and Options.<br>First character indicates the Transport protocol class.<br>Second character indicates the TPDU numbering format.<br>Third character indicates the explicit flow control (in class 2).<br>The five classes of Transport protocol are: |

| | |
|---|---|
| 0 | Simple class |
| 1 | Basic error recovery class |
| 2 | Multiplexing class |
| 3 | Error recovery and multiplexing class |
| 4 | Error detection and recovery class |

Class is listed only for the following TPDU's:

| | |
|---|---|
| CC | Connection Confirm |
| CR | Connection Request |

The valid values for the option are:

| | |
|---|---|
| E | For Extended |
| N | For Normal |
| F | Explicit flow control |

Option is listed only for the following TPDU's:

| | |
|---|---|
| CC | Connection Confirm |
| CR | Connection Request |
| AK | Acknowledgement |
| DT | Data |
| EA | Expedited Acknowledgement |
| ED | Expedited Data |

ECR

For the following TPDU's:

| | |
|---|---|
| CC | Connection Confirm |
| CR | Connection Request |

The first character is the use–of expedited data bit.
The second one is the use–of checksum bit.

The valid values are ”E” for expedited data and ”C” for checksum.

For the following TPDU's:

| | |
|---|---|
| DR | Disconnection Request |
| ER | Error |

This field indicates the reasons or reject codes.

Valid reason values (for DR) are:

| | |
|---|---|
| 0 | Reason not specified |
| 1 | Congestion at TSAP |
| 2 | Session entity not attached to TSAP |
| 3 | Address unknown |
| 128 | Normal disconnect initiated by session entity |
| 129 | Remote transport entity congestion at connect request time |
| 130 | Connection negotiation failed (proposed class not supported) |
| 131 | Duplicate source reference detected for the same pair of NSAPs. |
| 132 | Mismatched references |
| 133 | Protocol error |
| 134 | Not used |
| 135 | Reference overflow |
| 136 | Connection request refused on this network connection |

| | |
|---|---|
| 137 | Not used |
| 138 | Header or parameter length is invalid |
| other | Reason code value outside of ranges (0–3, 128–138) |

Valid reject values (for ER) are:

| | |
|---|---|
| 0 | Reason not specified |
| 1 | Invalid parameter code |
| 2 | Invalid TPDU type |
| 3 | Invalid parameter value |
| other | Reject cause outside of ranges (0–3) |

Opt The variable part of TPDU.

If a parameter is present, a single letter is listed for it.

(nn?) is listed in place of the letter if the parameter code is unknown, where nn is the hexadecimal value of the parameter code.

! is listed immediately after the letter if the parameter value is incorrect.

For example

```
c(47?)p!
```

Where 0x47 is an unknown code parameter and priority value is invalid.

The parameters displayed are:

| | |
|---|---|
| a | Alternate class |
| c | Checksum |
| d | Delay |
| f | Flow control confirmation |
| i | Inactivity timer |
| k | Acknowledgement time |
| n | Additional information |
| o | Additional options |
| p | Priority |
| q | Subsequence number |
| r | Residual error rate |
| t | Reassignment time |
| u | Throughput |
| v | Version number |
| w | Preferred maximum TPDU size |
| x | Calling TSAP–ID |
| y | Called TSAP–ID |
| z | TPDU size |

Concerning the ”z” parameter (TPDU size):

The value is listed considering the following codes:

| | |
|---|---|
| 7 | 128 bytes |
| 8 | 256 bytes |
| 9 | 512 bytes |
| 10 | 1024 bytes |
| 11 | 2048 bytes |
| 12 | 4096 bytes |
| 13 | 8192 bytes |

**COTP Layer – Extended Display**

ODIT presents a extended display of PDUs (COTP) which completes the summary display described above.

The following list gives all information that may appear under summary line, when extended display mode is selected.

For fixed part of PDU:

The TPDU type description is displayed.

For example :

| | |
|---|---|
| (CR) | Connection Request |

Reason = <reason text>

Reject cause = <reject cause text>

**COTP Layer – Extended Display**

For variable part of PDU:

| | |
|---|---|
| (a) | Alternate class = <value> |
| (c) | Checksum value = <value> |
| (d) | Transit Delay value = <value> |
| (f) | Flow control confirmation |
| (f) | Your sequence number = <value> |
| (f) | + Your subsequence number = <value> |
| (f) | + Your credit = <value> |
| (i) | Inactivity timer = <value> milliseconds |
| (k) | Acknowledgement time = <value> milliseconds |
| (n) | Additional information = <value> |
| (o) | Use of Request Acknowledgement in class 1,3,4 |
| (o) | Use of Selective Acknowledgement in class 1,3,4 |
| (o) | Use of Network Expedited in class 1 |
| (o) | Use of Receipt Confirmation in class 1 |
| (o) | Use of Checksum in class 4 |
| (o) | Use of Expedited Data |
| (p) | Priority value = <value> |
| (q) | Subsequence number = <value> |
| (r) | Residual error rate |
| (r) | + Target value = <value> power of 10 |
| (r) | + Minimum acceptable = <value> power of 10 |
| (r) | + TSDU size of interest = <value> power of 2 |
| (t) | Reassignment time = <value> seconds |
| (u) | Throughput value = <value> |
| (v) | Version number = <value> |
| (w) | Preferred maximum TPDU size = <value> |
| (x) | Calling TSAP–ID = <value> |
| (y) | Called TSAP–ID = <value> |
| (z) | TPDU size = <value> |

For errors on PDU:

| | |
|---|---|
| ??? | Error message = <Text of error> |

## How to Analyze COSP Layer Traces

### Access

From the Analyzer Output Options menu, select the field:

```
COSP layer
```

The following trace summary is displayed:

**COSP Layer**

| **Time** | **No** | **Size** | **Way** | **Type** | **Errs** | **LgH** | **CT** | **Data** | **PGIs and PIs** |
|---|---|---|---|---|---|---|---|---|---|
| 04:55.21.030 | 001 | 70 | ⇐ | AA | | 9 | 1 | 61 | |
| 04:55.33.060 | 002 | 70 | ⇒ | AB | | 12 | 1 | 58 | |
| 04:55.39.230 | 003 | 70 | ⇐ | AC | | 13 | 1 | 57 | |
| 04:55.48.340 | 004 | 70 | ⇒ | CN | | 7 | 1 | 63 | I[m]A[ozv]fxyU |
| 04:55.57.440 | 005 | 70 | ⇒ | DN | | 5 | 1 | 65 | |
| 04:56.13.550 | 006 | 70 | ⇐ | ?1B | PTYP? | | | | |

### Field descriptions

| | |
|---|---|
| Time | The time when the TSDU was transmitted to (or from) adjacent layer. |
| No | A running count of TSDU's. |
| Size | SPDU size in bytes. Decimal format, 5 digits.If the SPDU displayed is the first SPDU in a TSDU, this is the full TSDU size. If the SPDU is a concatenated SPDU, this is the remainding size of the TSDU. |
| Way | Incoming or Outgoing PDUs, with display format: |

| | |
|---|---|
| ⇒ | COSP PDU sent to remote station. |
| ⇐ | COSP PDU received from remote station. |

Type

Session SPDU's are listed below:

| | |
|---|---|
| AA | Abort accept |
| AB | Abort |
| AC | Accept |
| AD | Activity discard |
| ADA | Activity discard acknowledgement |
| AI | Activity interrupt |
| AIA | Activity interrupt acknowledgement |
| AR | Activity resume |
| AS | Activity start |
| CD | Capability data |
| CDA | Capability data acknowledgement |
| CDO | Connect data overflow |
| CN | Connect |
| DN | Disconnect |
| DT | Data transfer |
| ER | Exception report |
| ED | Exception data |
| EX | Expedited |
| FN | Finish |
| GT | Give token |
| GTA | Give token acknowledgement |
| GTC | Give token confirm |
| MIA | Minor synchronisation acknowledgement |

| | |
|---|---|
| MIP | Minor synchronisation point |
| MAA | Major synchronisation acknowledgement or Activity end ack |
| MAP | Major synchronisation point or Activity end |
| NF | Not finished |
| OA | Overflow Accept |
| PR | Prepare |
| PT | Please token |
| RA | Re synchronisation acknowledgement |
| RF | Refuse |
| RS | Re synchronisation |
| TD | Typed data |
| ?<code> | PDU type unknown |

Errs — Error message field, with possible values:

| | |
|---|---|
| BLPG? | Bad Length for PGi |
| BLPI? | Bad Length for PI |
| LPGP? | Length of Pi is Greater than Pgi |
| LPGS? | Length of Pi or pgi is Greater than Si |
| PPIV? | Pi or Pgi code is InValid |
| PPVI? | Pi or Pgi Value is Invalid |
| PTYP? | Pdu TYPe is invalid |

LgH — Length indicator value for this SPDU. Decimal format, 4 digits.

CT — Concatenation and Category.

Column 1 contains concatenation information:

| | |
|---|---|
| g | Give token. |
| p | Please token. |
| ” | SPDU is concatenated with SPDU displayed on the line above. |
| blank | First SPDU of a TSDU. |

Column 2 is the category information:

| | |
|---|---|
| 0 | Type 0 SPDU is printed. |
| 1 | Type 1 SPDU is printed. |
| 2 | Type 2 SPDU is printed. |

Data — Number of bytes of user data sent in this SPDU. Decimal format, 5 digits.

PGIs and PIs   The variable parameters within the Session PDU (SPDU).

If a parameter is present, a single letter is listed for it.

(nn?) is listed in place of the letter if the parameter code is unknown, where nn is the hexadecimal value of the parameter code.

! is listed immediately after the letter if the parameter length or the parameter value is incorrect.

For example

A(99?)v!

Where 0x99 is an unknown code parameter and version number is invalid.

The displayed PGI parameters are:

| | |
|---|---|
| A | Connect/Accept Item<br>If the length is greater than 0,<br>PIs are displayed between ”[]”. |
| E | Extended User Data No value is displayed. |
| I | Connection Identifier<br>If the length is greater than 0,<br>PIs are displayed between ”[]”. |
| L | Linking Information<br>If the length is greater than 0,<br>PIs are displayed between ”[]”. |
| U | User Data<br>No value is displayed.<br>Number of user data octets is included in the count of the ”Data”field. |

The displayed PI parameters are:

| | |
|---|---|
| a | Activity Identifier<br>No value is displayed. |
| b | Additional Reference Information<br>No value is displayed. |
| c | Common Reference<br>No value is displayed. |
| d | Transport Disconnect<br>The transport connection status is displayed:<br>K Transport connection to be kept<br>R Transport connection to be released<br>U User Abort<br>E Protocol Error<br>N No Reason<br>I Implementation restriction stated in the PICS. |
| e | Enclosure Item<br>The Session SDU is displayed:<br>B Beginning of Session SDU<br>E End of Session SDU |

f Session User Requirements
The functional unit is displayed if proposed:

| | |
|---|---|
| A | Activity Management |
| C | Capability Data |
| D | Duplex |
| E | Expedited Data |
| H | Half Duplex |
| I | Minor Synchronisation |
| J | Major Synchronisation |
| N | Negotiated Release |
| R | Re synchronisation |
| S | Symmetric synchronise fu |
| T | Typed Data |
| X | Exceptions |

g Re synchronisation Type Item
g2 Second Re synchronisation Type Item
The values displayed are:

| | |
|---|---|
| A | Abandon |
| R | Restart |
| S | Set |

h Synchronisation Type Item
No value is displayed.

i Initial Serial Number
i2 Second Initial Serial Number
No value is displayed.

k Token Setting Item
Four letters are displayed. In order, they represent the following tokens:
Release token
Major/activity token
Synchronisation minor token
Data token

The letters indicate the initial token positions:

| | |
|---|---|
| C | Called user's choice |
| I | Initiator's side |
| R | Responder's side |
| U | Undefined |

l Reflect Parameter Values
No value is displayed.

m Calling user reference
No value is displayed.

n Called user reference
No value is displayed.

o Protocol Options
The concatenation capability displayed is:

| | |
|---|---|
| X | Able to receive extended concatenation SPDUs. |

| | |
|---|---|
| p | Prepare Type<br>The display indicates what type of SPDU to prepare for:<br>A Major Synchronisation Acknowledgement<br>B Prepare for Abort PDU<br>K Re–synchronisation Acknowledgement<br>R Re–synchronisation |
| q | Data Overflow<br>No value is displayed. |
| r | Reason Code<br>The values displayed are:<br>0 Reason not specified<br>1 Rejection by responder due to congestion<br>2 Same as 1, user data follows<br>129 SSAP identifier<br>130 User not attached to SSAP<br>131 SPM congestion at connect time<br>132 Proposed protocol versions not supported<br>133 Rejection by the SPM; reason not specified<br>134 Rejection by the SPM; implementation restriction stated in the PICs. |
| s | Serial Number |
| s2 | Second Serial Number No value is displayed. |
| t | Token Item Tokens which are given are displayed:<br>D Data Token<br>M Major/activity Token<br>R Release Token<br>S Synchronisation–MinorToken |
| u | User Data<br>No value is displayed. |
| v | Version Number<br>Value displayed are:<br>1 Version 1<br>2 Version 2 |
| x | Calling Session Selector<br>No value is displayed. |
| y | Called Session Selector<br>No value is displayed. |
| z | TSDU maximum size<br>No value is displayed. |

**COSP Layer – Extended Display**

ODIT presents a extended display of PDUs (COTP) which completes the summary display described above.
The following list gives all information that may appear under summary line, when extended display mode is selected.

For SIs:

The SPDU type description is displayed.

For example:

| | |
|---|---|
| (MIA) | Minor Synchronisation Acknowledgement |

For PGIs:

| | |
|---|---|
| (A) | PGI: Connect/Accept Item |
| (E) | PGI: Extended User Data |
| (I) | PGI: Connection Identifier |
| (L) | PGI: Linking Information |
| (U) | PGI: User Data |

For PIs:

| | |
|---|---|
| (a) | PI: Activity Identifier |
| (b) | PI: Additional Reference Information |
| (c) | PI: Common Reference |
| (d) | PI: Transport Disconnect |
| (e) | PI: Enclosure Item |
| (f) | PI: Session User Requirements |
| (g) | PI: Re synchronisation Type Item |
| (g2) | PI: Second Re synchronisation Type Item |
| (h) | PI: Synchronisation Type Item |
| (i) | PI: Initial Serial Number |
| (i2) | PI: Second Initial Serial Number |
| (k) | PI: Token Setting Item |
| (l) | PI: Reflect Parameter Values |
| (m) | PI: Calling SS–user reference |
| (n) | PI: Called SS–user reference |
| (o) | PI: Protocol Options |
| (p) | PI: Prepare Type |
| (q) | PI: Data Overflow |
| (r) | PI: Reason Code |
| (s) | PI: Serial Number |
| (s2) | PI: Second Serial Number |
| (t) | PI: Token Item |
| (u) | PI: User Data |
| (v) | PI: Version Number |
| (x) | PI: Calling Session Selector |
| (y) | PI: Called Session Selector |
| (z) | PI: TSDU maximum size |

For errors on PDU:

| | |
|---|---|
| ??? | Error message = <Text of error> |

**NOTE**: Each Parameter value is decoded and displayed in a symbolic format under the according Parameter identifier (PI) line described above.

## Example of OSI Layer Trace Analysis

From the Output Options menu, select:

```
MAC Layer
LLC Layer
CLNP Layer
COTP Layer
COSP Layer
```

The following trace summary is displayed:

```
MAC Layer
Time          No    Size   Way   Type      LocMACaddr       RemMACaddr

10.06.52.29   002   188    ⇒     Ethernet  0x2608c2c8b79    0x020608c2fa32b


LLC Layer
Time          No    Size   Way   Type   Errs   Lsap   Rsap    Frame Type

10.06.52.29   002   174    ⇒     UI            fe     fe      Information


CLNP Layer
Time          No    Size   Way   Type   Errs LiT  SME DUI  SgOf   Data   Opt

10.06.52.29   002   171    ⇒     DT          10   YNY 99   0      139    cp


COTP Layer
Time          No  Size  Way  Type Errs  Loc   Rem    NR   EOT C  Data  CO  ECR Opt

10.06.52.29   002 139   ⇒    DT               0009   0    T      134


COSP Layer
Time          No    Size   Way   Type   Errs LgH  CT   Data   PGIs and PIs

10.06.52.29   002   134    ⇒     CN          132  1    100    I[m]A[ozv]fxyU
```

# Chapter 9. OSI Stack Diagnostic Toolkit

This chapter explains how to exploit the error codes and trace management facilities using the diagnostic and maintenance tools present in the OSI stack.

## Overview

The maintenance tools described in this section supply the system administrator with additional error code and trace management information not directly available through the System Management Interface Tool (SMIT).

You can find more information in:

- **trmad** Provides MAD driver trace and memory trace management facilities, see page 9-2.

  **Note:** A configuration must be active and super–user authority is required in order to use **trmad**.

- **pmaderror** : Provides error code parsing facilities, see page 9-11.
- **lookx25** Provides configuration information and statistics about physical, logical and network layer information for all X.25 adapters managed by the OSI Stack, see page 9-13.
- **osisnap** Provides stack problem data for transfer to removable media, see page 9-23.
- **osiping** : Sends a Transport Protocol Data Unit (TPDU) to obtain a Response from a host or a gateway, see page 9-26.

# trmad (MAD Driver Trace Management)

## Purpose

The **trmad** command is used to help the user activate trace operations on the OSI driver. These operations are performed only on the loaded configuration and are not stored in the ODM base.

This tool is mainly intended for use by **Technical Support teams**. Readers are advised to use ODIT for their trace operations.

Unlike ODIT, all trace settings are lost at the next stack de-activation/re-activation or system reboot.

## Syntax

**trmad**

## Description

All actions performed on the OSI Stack driver are traced in a trace buffer. **trmad** helps to tune the trace by:

- setting trace layers
- setting trace between modules
- setting trace between user space and kernel space
- setting trace buffer length
- editing trace buffer
- giving statistical information.

## Parameters

### First Screen: **MAD internal trace**

Configuration

```
exec file       : /usr/lib/drivers/osi/osi_frame
Memory file     : /dev/kmem

/usr/lib/drivers/osi/osi_frame          loaded
/usr/lib/drivers/pse/osi/osi_load       loaded
/usr/lib/drivers/pse/osi/cosp           not loaded
/usr/lib/drivers/pse/osi/copp           not loaded
```

This indicates the parts of the OSI stack currently loaded. Here, only lower layers are present.

## Main Menu

```
******************** Main Menu ********************************

        Total Memory used   =   566 Kb
        Trace Buffer Length =   128 Kb
        Nb OSI driver chan  =     2
        Nb Active TPI cnx   =     0

-----------------------------------------------------------------

 c) MAD trace Configuration    m) Memory analysis
 l) LAYERS trace setting       s) OSI Stack Statistics
 e) Edit OSI trace buffer (vi) r) Reset OSI trace buffer
 i) OSI driver Information     L) change OSI trace buffer Length

 q) quit
```

The upper part of the main menu screen provides the following information:

Total Memory used
: indicates all the kernel memory allocated dynamically by the OSI stack

Trace Buffer Length
: indicates the size of the circular trace buffer

Nb OSI driver chan
: indicates the number of open() on OSI driver

Nb Active TPI cnx
: indicates the number of open() on TPI interface

The lower part of the main menu screen displays the menu items and their associated keys described hereafter:

| | |
|---|---|
| **c** | Used to configure trace parameters and filters. See the MAD Trace Configuration menu on page 9-4. |
| **l** | Provides the trace level setting. See the Layers Trace Setting Menu on page 9-6. |
| **e** | The buffer trace is converted into ASCII as *ositrace.txt* file and vi is called. The variable OSI_TRACE_DIR is used to define the *ositrace.txt* file directory. If it is not defined, */var/tmp/osi/* is used. |
| **i** | Provides OSI stack layer information and status. See the OSI Driver Information Menu on page 9-7. |
| **m** | Displays OSI stack memory use. See the Memory Analysis Menu on page 9-10. |
| **s** | Provides OSI stack statistics on the major parameters.See “Accessing OSI Stack Information” on page 5-1. |
| **r** | Resets buffer trace, cleans all the information located in trace buffer. |
| **L** | Used to change the circular trace buffer length. |

## MAD Trace Configuration Menu

```
************* Main Menu > MAD trace Configuration **************

        PROTOCOL   : Set
        ICB        : Set
        UCB        : Set

        FILTERS ON MODULES:

As source:
As Dest  :
Always   :  LLC ADM TIM MAD MAC
UCB      :  ADM MAD

-----------------------------------------------------------------

  p) Set/Unset PROTOCOL Trace         t) OSI trace buffers tune
  i) Set/Unset ICB trace              m) OSI Driver trace level
  u) Set/Unset UCB trace              a) Set/Unset all Filters
  f) Enable filters on module         c) Set/Unset Complete trace

  q) Return previous menu
```

The upper part of the MAD Trace Configuration screen provides information on the state of the trace flags for Protocol, ICB and UCB trace and of the filters for the OSI stack, osi_low layers and trace messages.

The lower part of the MAD Trace Configuration screen displays the menu items and their associated keys described hereafter:

| | |
|---|---|
| **p** | Enables or disables protocol trace. When set, all the PDUs sent or received are traced, even on LAN, WAN or CLNS loopback. |
| **i** | The ICBs that are traced (or not) depend on the filter list. (ICB stands for Internal Control Block). |
| **u** | The UCBs that are traced (or not) depend on the filter list. (UCB stands for User Control Block). |
| **f** | Enables or disables Module in the filter list. |
| **t** | Modification of OSI Driver Trace Service. An example of display is provided next page. |
| **m** | Sets OSI driver interface trace levels. |
| **a** | Enables or disables all the filters on modules for ICB or UCB traces. |
| **c** | Enables or disables the complete trace. The Complete mode shows an absolute time stamp on each line and the non Complete shows only the delta from the previous line. By default the trace is not Complete. |

**Example** of display on **t** key selection (Modification of OSI Driver Trace Service):

```
                Modification OSI Driver Trace Service :

ICB/UCB chain truncature value=  64 (32 ->  255):

Stream Message trace truncature value= 256 (32 -> 1024):

PDU truncature for protocol trace= 128 (64 -> 1536):

Protocol Analysis:                    (Strike 0 to disable display)
        MAC Analysis       (1):
        LLC Analysis       (1):
        X253 Analysis      (1):
        ESIS Analysis      (1):
        CLNP Analysis      (1):
        COTP Analysis      (1):
        COSP Analysis      (1):
```

## Layers Trace Setting Menu

```
************ Main Menu > LAYERS trace setting *****************

                          1 1 1 1 1 1 1 1 1 1 2 2 2 2 2 2 2 2 2 2 3 3 3
----- 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1 2
ACSE  X . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . .
COPP  X . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . .
COSP  X . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . .
MSDSG X . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . .
TPI   . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . .
COTP  X X . . . . . .
CLNP0 . . . . . . . .
    1 . . . . . . . .
    2 . . . . . . . .
    3 . . . . . . . .

-----------------------------------------------------------------------------

  n) CLNS Layer  (IP)                     x) COTP Driver (TPI)
  t) COTP Layer  (Transport)              s) COSP Layer  (Session)
  p) COPP Layer  (Presentation)           a) ACSE Layer  (ACSE)
  m) MSDSG Layer                          s) Security Module
  q) Return previous menu
```

The upper part of the Layers Trace Setting screen displays a summary of the OSI layers trace settings.

The lower part of the Layers Trace Setting screen displays the menu items and their associated keys.

The use of this menu is for experts only. It is advised to the user who needs setting traces to use ODIT with the more friendly SMIT interface. See OSI Stack Layers Trace Managementon page 7-1.

## OSI Driver Information Menu

```
************ Main Menu > OSI driver Information   ***************

        ES/IS Type          = ES
        Lg Static RIB       = 256
        Lg Dynamic RIB      = 256

--------------------------------------------------------

  n) local NSAPs List       b) Static & Dynamic RIB List
  m) defined Modules        o) MAD active Contexts
  s) MAD active Schedulers  t) COTP layer counters
  l) LLC statistics         c) Cnx licence status
  p) Pending timer info
  q) Return previous menu

        ---->
```

The upper part of the OSI Driver Information screen displays the ES/IS mode and the length of the RIB tables.

The lower part of the OSI Driver Information screen displays the menu items and their associated keys described hereafter:

**n** Displays the list of the local NSAPs. For example:

```
         ---- Local NSAPs Table (32)----

        1 490099999999
        2 590099999999
```

**m** Displays the list of OSI driver internal declared modules. For example:

```
+-------+----+-----+--------+--------- FIFO ---------+---------+
        + Name + No + UCB + Enable + Pend | Maxi |   Total  +
        +------+----+-----+--------+------|------|----------+
        +  LLC + 02 +     +   Y    +    0 |    0 |        0 |
        +  X3  + 03 +     +   Y    +    0 |    0 |        0 |
        +  TRA + 04 +     +   Y    +    0 |    0 |        0 |
        +  ADM + 07 +  Y  +   Y    +    0 |    0 |        0 |
        +  MAD + 09 +  Y  +   Y    +    0 |    0 |        0 |
        +  IP  + 0f +     +   Y    +    0 |    0 |        0 |
        +  MSD + 10 +     +   Y    +    0 |    0 |        0 |
        +  MAC + 12 +     +   Y    +    0 |    0 |        0 |
        +  X2  + 20 +     +   Y    +    0 |    0 |        0 |
        +  mX3 + 36 +  Y  +   Y    +    0 |    0 |        0 |
        +  ANN + 38 +     +   Y    +    0 |    0 |        0 |
        +  mTR + 46 +  Y  +   Y    +    0 |    0 |        0 |
        +  TPI + 47 +     +   Y    +    0 |    0 |        0 |
        +  aSe + c5 +     +   Y    +    0 |    0 |        0 |
        +  aPr + c6 +     +   Y    +    0 |    0 |        0 |
        +  aAC + c7 +     +   Y    +    0 |    0 |        0 |
        +  aNL + ee +  Y  +   Y    +    0 |    0 |        0 |
        +  com + ff +     +   Y    +    0 |    0 |        0 |
        +------+----+-----+--------+------------------------+
```

**s** Displays the list of schedulers and statistics.

```
+---+-----+---+--------+----+------+------+-----------+------+---
|                            |       FIFO  |     TIMERS          |
| N | St  | W |  pid   | Nm | Pend | maxi |Total|Alloc|nb| maxi |
+---+-----+---+--------+----+------+------+-----------+------+---
| 0 | Run |   |   1661 | 13 |    0 |    6 |1930 | 512 |2 |    2 |
|IT |                        |    0 |    0 |      2048 |
|Mod| LLC TRA ADM MAD IP  MAC ANN mTR TPI aSe aPr aAC com
+---+-----+---+--------+----+------+------+-----------+------+---
```

**l** Displays LLC statistics. The following screen appears in scan mode. CTRL C must be entered to stop the display.

```
ETH: 9.0.99.0.99.99 nbPending= 0
        0 -> RefUp= 10 nbNotAck=  3
        1 -> RefUp= 10 nbNotAck=  1
        3 -> RefUp= 10 nbNotAck=  3
```

**p** Displays all the active timers in the stack:

```
Timer scheduler[0] status=1
[@0x06df17e8]  1’ 0.436 IP  ctx=0x06de27f0 id=0x1234
[@0x06df17cc]  2’59.148 ANN ctx=0x06de2868 id=0x0336
```

**b** Displays the contents of static and dynamic RIB in the stack memory.

**o** Displays the list of opened MAD with statistics.

```
+----------+-------------+---------+---+---|    FIFO            |
|  Channel |     AGE     |   Pid   | W | S | Pend | Maxi |Total |
+----------+-------------+---------+---+---+------+------+------+
| 05b5d020 |  23h  5’ 47 |    6309 | W |   |    0 |    1 |    1 |
| 05b5d25e |   1h 10’ 47 |    1987 |   | S |    0 |  101 |10034 |
```

**t** The following screen appears in scan mode. It displays COTP layer counters illustrating COTP activity; CTRL C must be entered to stop the display.

```
================================================================
               ***   TRANSPORT VARIABLES  ***
================================================================
    No alarm
================================================================
                     NbActCon ... 0 (Max 512)
              NbDataOctetsSent ... 688700
               NbDataOctetsRec ... 688700
                     NBPDUSent ... 9633
                      NBPDURec ... 9632
                  NBPDURetrans ... 0
  NbLocInSuccsCon ... 48   NbRemInSuccsCon ... 48
 NbLocInUnsccsCon ...  1   NbRemInUnsccsCon ... 1
     NbLocErrDisc ..  50       NbRemErrDisc ... 0
  NbPDUVoidConRec ...  1        BPDUProtErr ... 0
       NBTimInact .    2      NBCrPDUConges ... 0
 NBCrPDUCfgErrDet .    1  NBRefuCrPDUCfgErr ... 1
NBCrPDUProtErrDet ...  2          NBTimCrPDU ... 0
  NBDetPDUProtErr ...  0   NBRefuPDUProtErr ... 2
 NBDisdPDUCksFail ...  0            TimAck ... 10  DefMaxRetrans ...    6
  NBMaxRetransTim ...  0  TimLocRetrans ... 10000   TimInactLAN ... 70000
 NbCreditZeroSent ...  0       TimWindow .. 60000   TimInactLAN ...230000
================================================================
```

**c** Displays the licence status of the installed packages:

```
Low Layer       : Not Locked
Upper Layer     : Not Locked
Max3 Layer      : Not Locked
PAD             : Not Available
BASIC COM       : Not Available

Layers Statistics
- - - - - - - - - -

General         all=  0/2048    in=  0/2048     out=  0/2048
COTS            all=  0/2048    in=  0/2048     out=  0/2048
MAX3            all=  0/2048    in=  0/2048     out=  0/2048

- - - - Strike Return to continue - - - -
```

## Memory Analysis Menu

```
*************** Main Menu > Memory analysis ********************

Maximum memory :                 17408 Kbytes
Maximum Memory allocated :        1044 Kbytes
Memory currently allocated :       420 Kbytes

-----------------------------------------------------------------
------------

  m) Memory statistics         a) MAD Alloc list of buffers
  c) Count chain               l) List chain by owner
  v) Display a chain           p) Dump used element in pool
  t) GM trace tuning           d) Dump memory

  q) Return previous menu
```

# pmaderror (Error Code Parsing)

## Purpose

The **pmaderror** command is used for parsing an 8–digit OSI stack error code using files found in the directory */usr/include/osi/err* and also for parsing ASCII text associated with an *errno* value.

Error codes are given in hexadecimal format and *errno* values are given in decimal format.

For an error or diagnostic code, the stack displays the following information:

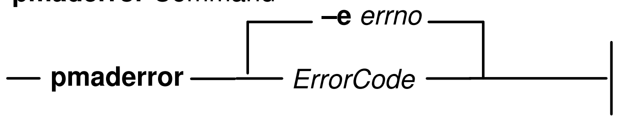
- Module = component where the error originates from
- Func/Cause = context in which the error appears
- Error/Diag = diagnostic message

For further information, refer to “Error and Diagnostic Codes Analysis” on page 2-3.

## Syntax

### Unix Command

**pmaderror** Command



where **-e** is a decimal value and *ErrorCode* is an 8-byte hexadecimal value.

### Library Function

The **pmaderror( )** library function is located in */usr/lib/osi/libosi_trace.a*. It is used for the conversion of error codes. The syntax is as follows:

```
int pmaderror (unsigned long ErrCode, char ** pAsciiModule,
        char ** pAsciiCause, char ** pAsciiError)
```

The possible return codes are:

1 – indicating the operation was successful;

-1 – indicating the ASCII error message catalog is not present;

0 – indicating the operation failed.

**Note:** The non–NULL pAscii pointers must be freed by using the **free( )** function.

## Examples

The command:

**pmaderror 0401041a**

gives the following display:

```
----------------  MAD error 0x0401041a -------------------
| Module       : transport layer                          |
| Func / Cause :                                          |
| Error / Diag : Unknown local NSAP                       |
----------------------------------------------------------
```

The command:

**pmaderror -e 205 -e 4**

gives the following display:

```
----------------  errno 205 -------------------------------
| MAD  (205) : OSI driver out of order                    |
-----------------------------------------------------------
----------------  errno 4 ---------------------------------
| System     : Interrupted system call                    |
-----------------------------------------------------------
```

Below is an example of use of the **pmaderror( )** function:

```
DisplayAscii_OsiStackError( unsigned long ErrCode)
{
char * perr, *pset, *pmod;
int    ret;

       ret = pmaderror( ErrCode, &pmod, &pset, &perr);

       switch ( ret ) {
               case 1 : printf("ErrCode 0x%08x : %s", ErrCode, perr);
                      if ( pmod ) free(pmod);
                      if ( pset ) free(pset);
                      if ( perr ) free(perr);
                      break;

               case 0 : fprintf(stderr, "Error Code not found\n");
                      break;

               case -1 : fprintf(stderr, "Catalog File not found\n");
                       break;
         }
}
```

## lookx25

### Purpose

This tool provides configuration information and statistics about physical, logical, and network layer information for all X25 adapters managed by the OSI stack.

List of screens:

- Network Layer Configuration on page 9-14.
- Network Layer Statistics on page 9-16.
- Virtual Circuit Statistics on page 9-18.
- Link Layer Configuration on page 9-20.
- Physical Layer Configuration on page 9-22.

The screens displaying statistics are continuously refreshed.

### Syntax

**lookx25**

### Information Common to all Screens

#### Board on slot

MCA bus slot number into which the board is inserted.

#### Line

For the multi–port board X.25 High Performance Adapter, one of the 4 lines on the board (numbered 0 to 3).

## Network Layer Configuration

At start–off of command lookx25, the first screen displayed concerns X25 network layer configuration :

```
                              X25 LINE MONITOR

Board on slot : 5                                              Line : 0
————————————————————NETWORK LAYER (X25–3) CONFIGURATION————————————————————|
|                                                                           |
|          Subscription Address : 0x111          Total number of Vcs : 256   |
|                                                                           |
| NB PVC: 0                   MODE : DTE        FAST SELECTION: Y           |
|                                                                           |
| NB IN–ONLY  SVC: 0          FIRST IN–ONLY  SVC NB: 0     FLUX CONTROL: Y   |
| NB OUT–ONLY SVC: 0          FIRST OUT–ONLY SVC NB: 0     DFLT PACKET SIZE: 128 |
| NB TWO WAYS SVC: 256        FIRST TWO WAYS SVC NB: 1     DFLT WINDOW SIZE: 2   |
|                                                                           |
| TO RESTART: 180   TO CALL: 200   TO CLEAR: 180   TO WINDOW: 180   TO TRANS: 180 |
—————————————————————————————————————————————————————————————————————————————
| 1 – Next Line                                 3 – X25–2 Configuration     |
| 2 – Channel Statistics                        4 – X25–1 Configuration     |
| 5 – Exit                                      Your Choice –> ?            |
—————————————————————————————————————————————————————————————————————————————
```

### Number of PVC

Maximum nb of pvc configured.

### Mode

DTE: Data terminal equipment behavior.

DCE: Data computing equipment behavior.

### Fast selection

Y : Accepts incoming calls with fast selection parameters.

N : Does not.

### Number of in-only svc

Maximum number of in-only svc configured.

### First in-only svc

VC number of first in-only svc.

### Number of out-only svc

Maximum number of out-only svc configured.

### First out-only svc

VC number of first out-only svc.

### Number of two-way svc

Maximum number of two-way svc configured.

**First two-way svc**
VC number of first two-way svc.

**Flow control**
Y : Accepts to negotiate flow control parameters.

N : Does not.

**Default packet size**
Default size of the unit of information for X.25 packet level.

**Default window size**
Defines the number of (modulo) ordered consecutive packets authorized to cross the interface.

**Restart timeout**
ISO 8208 defined T20 timer.

**Call timeout**
ISO 8208 defined T21 timer.

**Clear timeout**
ISO 8208 defined T23 timer.

**Window timeout**
ISO 8208 defined T24 timer.

## Network Layer Statistics

The choice 2,“command=channel statistics”, in the screen “Network Layer Configuration”, displays the following sub-menu:

```
                                  X25 LINE MONITOR

Board on slot : 5                                                    Line : 0
─────────────────────────NETWORK LAYER  (X25–3) STATISTICS─────────────────────
|                                                                               |
|                                                Total number of Vcs  : 256     |
| Line State : OPENED VC                         ADM State : UNLOCKED           |
| Subscription Address : 0x111                                                  |
| DATA RX : 117        CALLS IN  : 4        RESTART IN : 0                      |
| DATA TX : 117        CALLS OUT : 10       RESTART OUT : 0                     |
|                                                                               |
|                                                                               |
|                               CV  STATUS                                      |
|                                                                               |
|                                         Data                 Resets            |
|  VC#      State         Remote Address  RX         TX        IN        OUT     |
─────────────────────────────────────────────────────────────────────────────────
| 5         TRANSFERT          111        11         0         0         0       |
| 6         TRANSFERT          111        10         0         0         0       |
| 256       TRANSFERT          555        0          26        0         0       |
| 7         TRANSFERT          111        10         0         0         0       |
─────────────────────────────────────────────────────────────────────────────────
| 1 – Return To Previous Screen           3 – Show 4 Previous VCs               |
| 2 – VC Statistics                       4 – Show 4 Next Vcs                   |
| 5 – Exit                                Your Choice –> 4                      |
─────────────────────────────────────────────────────────────────────────────────
```

## Line state

Opened line: network layer is in ready state, but no VCs are open.

Closed line: network layer is down.

Opened VC: network layer is in ready state, and VCs are open.

All opened VC: network layer is in ready state, and the number of opened VCs has reached the total nb of VCs for which the line is configured.

## Open VC

Number of virtual circuits currently open.

## Total number of VCs

Maximum number of virtual circuits that can be opened on the line identified by the board number and line number.

**Subscription Address**

The X25 line in an X25 network is identified by a number called the network user address (NUA) or subscription address.

Most public networks use the X.121 addressing standard to create NUAs.

**Administrative state**

Unlocked: the general module for administration (MAG) is ready to consult the ISO communication layers.

**Data received**

Number of PDUs received.

**Data transferred**

Number of PDUs transferred.

**Calls in**

Number of incoming calls detected.

**Calls out**

Number of outgoing calls detected.

**Restart in**

Number of restart packets received.

**Restart out**

Number of restart packets emitted.

## Virtual Circuit Statistics

The choice 2, “command=Channel Statistics”, in the screen “Network Layer Statistics”, displays the following sub-menu (specific to the consulted virtual circuit):

```
Board on slot : 5                                                        Line : 0
 ---------------------------------VIRTUAL CIRCUIT STATISTICS--------------------------
|          VC : 256                                  Type VC     : SWITCHED          |
|          Initiator : Yes                                                           |
|                                                                                    |
| Called Address              : 0x555                VC State  : TRANSFERT           |
| Calling Address             : 0x111                VC Owner : 0x3610               |
| Reception Window Size       : 2                    Transmission Window Size  : 2   |
| Reception Packet Size       : 128                  Transmission Packet Size  : 128 |
| Nb Data Packet Received     : 0                    Nb Data Packet Sent    : 4260   |
| Nb Octets Received          : 0                    Nb Octets Sent         : 545280 |
| Nb Reset Packet Received    : 0                    Nb Reset Packet Sent   : 0      |
| Nb RR Packet Received       : 0                    Nb RR Packet Sent      : 0      |
| Nb INT Packet Received      : 0                    Nb INT Packet Sent     : 0      |
|                                                                                    |
 ------------------------------------------------------------------------------------
| 1 – Return To Previous Screen 3 – Previous VC                                      |
| 2 – VC Number                                                4 – Next VC           |
| 5 – Exit                                                     Your Choice –> 4      |
 ------------------------------------------------------------------------------------
```

### VC#

The number of the virtual circuit.

### Called Address

The called address in the call request packet.

### Calling Address

The calling address in the call request packet.

### VC Type

Switched for a Switched Virtual Circuit (SVC).

Permanent for a Permanent Virtual Circuit (PVC).

### Initiator

Yes, when the call request packet was initiated.

No, when an incoming call packet was received.

### VC State

The states displayed concern the packet layer as defined in document ISO 8208:

| | |
|---|---|
| 1 | Restart states (r1, r2, r3) |
| 2 | Call setup and call clearing states (p1, p2, p3, p4, p5, p6, and p7) |
| 3 | Data transfer states (d1, d2, d3) |

Pret: state P1

Appel envoye: state P2

Appel recu: state P3

Paq_cl: state P6

P7: dxe clear indication

Transfer: state P4 or D1

D2: dte reset request

D3 dxe reset indication

R2: dte restart indication

R3: dxe restart indication

P5: call collision

## VC Owner

The subsequent protocol identifier (SPI) of the OSI module that owns the virtual circuit.

Refer to the Bull DPX/20 *OSI Services Reference Manual* for a complete list.

## Reception Window Size

Transmission window size

- Default window size,
- Or negotiated window size for the duration of a virtual call. Negotiation takes place if a facility request is inserted in the call packet.

## Link Layer Configuration

The choice 3, “command=X25–2 Configuration”, in the screen “Network Layer Configuration”, displays the following sub-menu:

```
                              X25 LINE MONITOR

Board on slot : 5                                                    Line : 0
 ————————————————————LINK LAYER (X25–2) CONFIGURATION—————————————————————
|                                                                          |
| Line State            : OPENED LINE        Automatic Start     : YES     |
| Type of Line          : HDLC               Delay to Restart    : 0       |
|  Protocol             : LAPB               PDU Sent Size       : 133     |
| Type of Local Station : DTE                PDU Rec Size        : 133     |
| Type of Remote Station : DCE                                             |
|                                                                          |
| Ctrl Frame Sent       : 0                  Ctrl Frame Rec      : 36970   |
| Info Frame Sent       : 0                  Info Frame Rec      : 0       |
| Nb Octets Sent        : 0                  Nb Octets Rec       : 0       |
| Reject Frame Sent     : 0                  Reject Frame Rec    : 0       |
| RNR Frame Sent        : 0                  RNR Frame Rec       : 0       |
| Nb PDU Re–Sent        : 0                  Nb PDU Error Rec    : 0       |
|                             Commands                                     |
| 1 – Return To Previous Screen                                            |
| 2 – Exit                                          Your Choice –> ?       |
 ——————————————————————————————————————————————————————————————————————————
```

### Line State

Opened line: link layer is in ready state.

Closed line: link layer is down.

### Type of Line

The link level is based on high-level data-link (HDLC). This procedure is synchronous and full duplex.

In HDLC, all commands, responses, and data are transmitted in frames.

### Protocol

The link access procedure is balanced (LAPB).

### Type of Local Station

DCE: Data Circuit–terminating Equipment.

DTE: Data Terminal Equipment.

### Ctrl Frame Sent

The number of control frames sent. This corresponds to supervisor and unnumbered frames.

### Ctrl Frame Rec

The number of control frames received. This corresponds to supervisor and unnumbered frames.

**Info Frame Sent**
Total number of info frames sent correctly.

**Info Frame Rec**
Total number of info frames received correctly.

**Nb Octets Sent**
Total number of user octets sent.

**Nb Octets Rec**
Total number of user octets received.

**Reject Frame Sent**
The number of supervision rejected frames sent.

This frame is used to ask for retransmission of frames starting off with number N(R) (reception sequence nb).

**Reject Frame Rec**
The number of supervision rejected frames received.

**RNR Frame Sent**
Total number of  Receive Not Ready (RNR) frames sent. This type of frame indicates that either the ETTD or ETCD is temporarly unable to accept other frames.

**RNR Frame Rec**
Total number of  Receive Not Ready (RNR) frames received.

**Nb PDU Re–Sent**
Total number of PDUs resent.

**Nb PDU Error Rec**
Reasons for an error PDU :

frame not delimited by two flags,

frame with an incorrect FCS,

frame with wrong number of bits between the flags,

frame with wrong address.

## Physical Layer Configuration

The choice 4, “command=X25–1 Configuration” in the screen “Network Layer Configuration”, displays the following sub-menu:

```
                              X25 LINE MONITOR

Board on slot : 5                                                    Line : 0

 ————————————————PHYSICAL LAYER (X25–1) CONFIGURATION——————————————————
| Line State          : OPENED LINE                                       |
| Speed               : 0                                                 |
 ——————————————————————————————————————————————————————————————————————
| Underruns Sent      : 0                                                 |
| Overruns  Rec       : 0                                                 |
| Abort Sequence Rec  : 0                                                 |
 ——————————————————————————————————————————————————————————————————————
|                              Commands                                   |
 ——————————————————————————————————————————————————————————————————————
| 1– Return To Previous Screen                                            |
| 2– Exit                                          Your Choice–> ?        |
 ——————————————————————————————————————————————————————————————————————
```

### Line state

Opened line: physical layer is in ready state.

Closed line: physical layer is down.

### Speed

Line speed in bits/sec.

Significant if the adapter generates the clock signal (possible only with X.25 High Performance Adapter).

### Underruns

Total number of underruns detected.

### Overruns

Total number of overruns detected by interface.

## osisnap Command

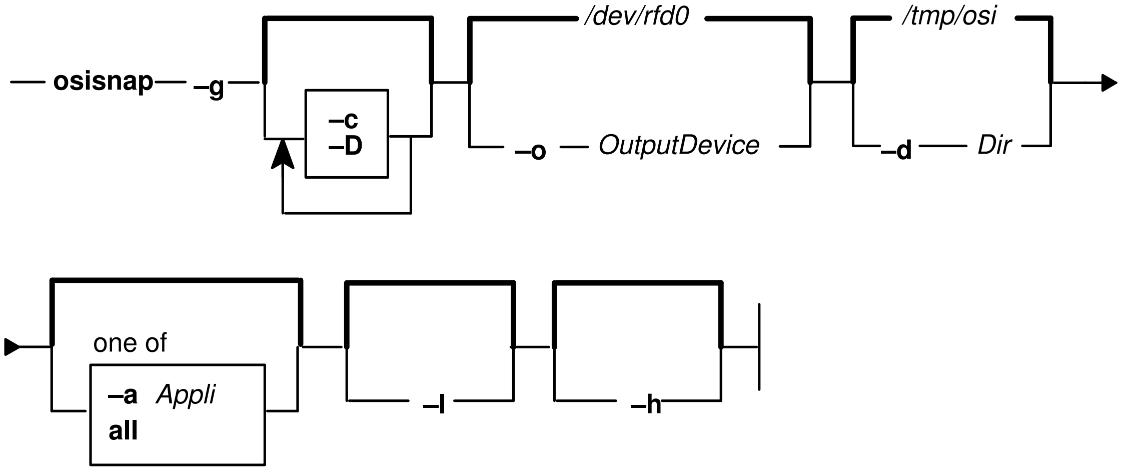
### Purpose

The **osisnap** command gathers information neccessary to report a problem encountered with the OSI stack and optionally puts this data on removable media.

### Syntax

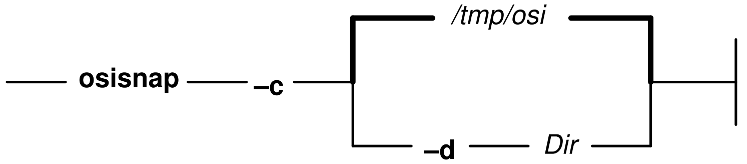
#### How to Gather Information and Dump it in a File

**osisnap** Command



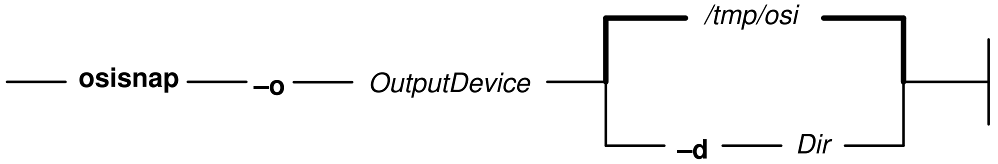
#### How to Create the *osisnap.tar.Z* File in a Target Directory

**osisnap** Command



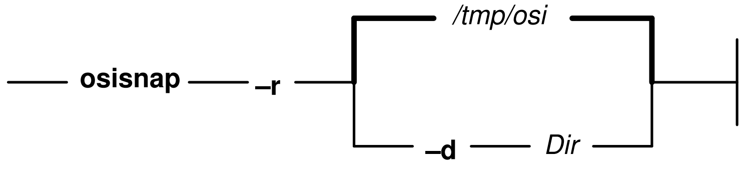
#### How to Send Information to a Removable Output Device

**osisnap** Command



### How to Remove Files from a Target Directory

**osisnap** Command



## Description

The **osisnap** command can gather useful information to be reported to technical support when a problem occurs in the OSI stack. Basically, **osisnap** collects information about the OSI configuration and then calls the **snap** command to collect more information about the machine configuration.

All files in the trace directory are copied in the snap. If the OSI_TRACE_DIR environment is set, it is used as the trace directory, if not, */var/tmp/osi* is used.

One of the **–c**, **–g**, **–o**, or **–r** flags must be specified; however, the **–r** flag cannot be specified with the **–c**, **–g** or **–o** flags.

## Flags

| | |
|---|---|
| **–h** | Provides help. |
| **–c** | Creates a compressed file *(osisnap.tar.Z)* in the target directory. If the **–g** flag is not specified, **osisnap** tries to compress any files found in the target directory. |
| **–d** | Specifies the directory in which the information is to be placed. Default */tmp/osi*. |
| **–D** | Gathers dump and UNIX information. The flag is passed to the **snap** command. |
| **–g** | Gathers OSI and general information. If a previous **osisnap** command has created files in the target directory, the user is asked if the directory must be purged before processing. |
| **–l** | Lists available application snaps. |
| **–o** | Sends information to the removable output device. If **–c** is specified, **osisnap** sends the compressed file *osisnap.tar.Z* to the output device, otherwise **osisnap** sends any files found in the target directory. If **–g** is not specified, **osisnap** sends any files found in the target directory to the output device. |
| **–r** | Removes files found in the target directory created by a previous **osisnap** command. |
| **–a** | Enables the user to obtain a snap for a particular application. |
| **all** | Enables to obtain snaps for all the applications recognized by the system. |

## Parameters

| | |
|---|---|
| *Dir* | The name of the target directory. The default directory is */tmp/osi*. |
| *OutputDevice* | The name of the removable output device. The default is */dev/rfd0*. |
| *Appli* | The name of an OSI application using the OSI stack (use **–l** to list). |

## Example

Create a compressed file containing OSI and general information and send it to the removable diskette media:

```
osisnap –gco /dev/fd0

osisnap –gco –a ftam
```

## Implementation Specifics

Part of the OSI framework delivery.

## File

| | |
|---|---|
| */usr/bin/osisnap* | Specifies the command file. |

## osiping Command

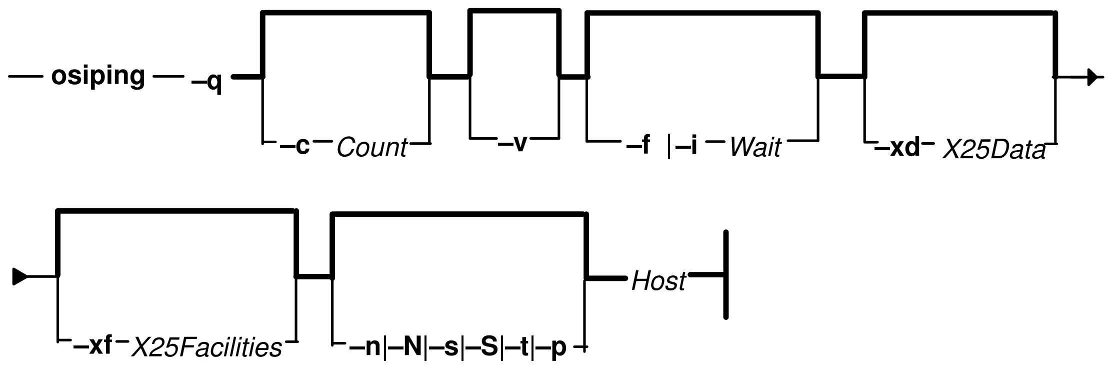
### Purpose

The **osiping** command sends a request to a remote OSI stack over Connection Oriented Transport Protocol (COTP) host using a network address. This address is composed either of a Network Service Access Point (NSAP) address or of a Sub Network Point of Attachment (SNPA) address, depending on the OSI stack profile used, the type of protocol – Full or Null ConnectionLess (CLNP), Connection Oriented (CONP) – and the type of network, LAN or WAN.

### Syntax

**osiping** [**–q**] [**–c** *Count* ] [ **–v** ] [ **–f** | **–i** *Wait* ] [ **–xd** *X25Data* ] [ **–xf** *X25Facilities* ] [ **–n** | **–N** | **–s** | **–S** | **–t** | **–p** ] *RemoteHost* [ *LocalHost* ]

**osiping** Command



### Description

The **osiping** command sends a Transport Protocol Data Unit (TPDU) to obtain a Response from a host or a gateway.

Several OSI stack addressing profiles can be used:

- Transport layer on Full CLNP, NSAP addressing on LAN or WAN,
- Transport layer on Null CLNP, NSAP addressing on LAN,
- Transport layer on CONS, NSAP addressing on WAN,
- Transport layer on Null CLNP, SNPA addressing on LAN or WAN.

In all these cases, class 4 is used on the Local Transport Layer.

This command is useful for:

- Determining the status of the network and various foreign hosts,
- Tracking and isolating hardware and software problems,
- Testing and measuring networks.

The command sends one Connect Request (CR), waits for a Connect Confirm (CC) or a Disconnect Request (DR) from the remote Transport. One line is output for every response received. One second separates each call.

The command calculates round–trip times and displays a brief summary on completion. It completes when the program times out or on receipt of a SIGINT signal.

The default is to continuously send CR until an interrupt is received (Ctrl–C).

Any local errors stop the command and a clear explanation of the error is displayed.

**CAUTION:**
**This tool does not need a remote responder. It works correctly in front of a Bull OSI stack and may work in front of another OSI stack.**

## Flags

| | |
|---|---|
| **–q** | Specifies quiet output. Nothing is displayed except the summary lines at startup time and when finished. |
| **–c** Count | Specifies that the number of requests specified by the Count variable are to be sent (and received). |
| **–v** | Verbose Mode, explaining Responses received or No Responses. |
| **–f** | Specifies the **flood–ping option**. The –f flag ”floods” or outputs CR as fast as they come back. **Note:** the **–f** flag is incompatible with the **–i** Wait flag. **For root user only**. |
| **–i** Wait | Waits the number of seconds specified by the Wait variable between the sending of each packet. The default is to wait for one second between each packet. **Note:** the **–i** Wait flag is incompatible with the **–f** flag. |
| **–xd** X25Data | Specifies X25 Data given as a hexadecimal buffer. |
| **–xf** X25Facilities | Specifies X25 Facilities as a hexadecimal buffer (Max 109 Bytes). |
| **–n** | Specifies a COTP OSI profile used: Full CLNP on LAN or WAN RemoteHost indicates an NSAP remote address. |
| **–N** | Specifies a COTP OSI profile used: Full CLNP on LAN, CONS on WAN RemoteHost indicates an NSAP remote address. |
| **–s** | Specifies a COTP OSI profile used: COTP on Null CLNP on LAN or WAN RemoteHost indicates an SNPA remote address (LSAP 0x20). |
| **–S** | Specifies a COTP OSI profile used: COTP on Null CLNP (LSAP 0xfe) RemoteHost indicates an SNPA remote address. |
| **–t** | Specifies a TP 1006 profile used. RemoteHost indicates a TCP/IP host internet name or address. |
| **–p** | Specifies a COTP OSI profile used: PVC access on WAN RemoteHost indicates a PVC name. |
| | **Note:** flag default value **–n** with Host indicates an NSAP. |
| **RemoteHost** | Depending on the options, represents: |
| | NSAP remote address with **–n**, **–N** option |
| | SNPA remote address with **–s**, **–S** option |
| | Internet name or address with **–t** option |
| | PVC name with **–p** option. |
| **LocalHost** | Allowed to connect under a specified local address: |
| | NSAP local address with **–n**, **–N** option |
| | SNPA local address with **–s**, **–S** option |
| | Irrelevant in the case of PVC. |

## Examples

1. To obtain information about host 49030508, enter:

   ```
   osiping 49030508
   ```

   Information similar to the following is displayed:

   ```
   OSIPING 49030508 (”I...”):
   to 49030508 (”I...”): seq=0 time=6 ms
   to 49030508 (”I...”): seq=1 time=13 ms
   to 49030508 (”I...”): seq=2 time=6 ms
   to 49030508 (”I...”): seq=3 time=8 ms
   \^C
   ---- OSIPING 49030508 (”I...”) Statistics ----
   4 CR transmitted, 4 responses, 0% loss
   round-trip min/avg/max = 6/8/13 ms
   ```

   **Note:** The output is repeated until an interrupt (Ctrl–C) is received.

2. To obtain information about host 02608c2fa32b, enter:

   ```
   osiping -s 02608c2fa32b
   ```

   Information similar to the following is displayed:

   ```
   OSIPING 02608c2fa32b:
   from Ethernet 02608c2fa32e to 02608c2fa32b: seq=0 time=6 ms
   from Ethernet 02608c2fa32e to 02608c2fa32b: seq=1 time=13 ms
   from Ethernet 02608c2fa32e to 02608c2fa32b: seq=2 time=6 ms
   from Ethernet 02608c2fa32e to 02608c2fa32b: seq=3 time=8 ms
   \^C
   ---- OSIPING 02608c2fa32b Statistics ----
   100 CR transmitted Through Ethernet 02608c2fa32e, 100 responses,
   0% loss round-trip min/avg/max = 6/8/13 ms
   ```

   **Note:** The output is repeated until an interrupt (Ctrl–C) is received.

3. To specify quiet output, enter:

   ```
   osiping -q 49030508
   ```

   Only summary information, similar to the following is displayed:

   ```
   OSIPING 49030508 (”I...”):
   \^C
   ---- OSIPING 49030508 (”I...”) Statistics ----
   100 CR transmitted, 100 responses, 0% loss
   round-trip min/avg/max = 6/8/13 ms
   ```

   **Note:** Although not displayed, the output of packets continues until an interrupt (Ctrl–C) is received.

4. To specify verbose output, enter:

   ```
   osiping -v 49030508
   ```

Information, similar to the following is displayed:

```
OSIPING 49030508 (”I...”):
COTP CR Timeout: seq=0 time=60 s
COTP CR Timeout: seq=1 time=60 s
COTP CR Timeout: seq=2 time=60 s
X253 Refuse by remote: seq=3 time=6 ms
X253 Refuse by remote: seq=4 time=6 ms
\^C
---- OSIPING 49030508 (”I...”) Statistics ----
5 CR transmitted, 5 responses, 0% loss
round-trip min/avg/max = 0/0/0 ms
```

**Note:** Although not displayed, the output of packets continues until an interrupt (Ctrl–C) is received.

**Error Messages**

”osiping: Invalid option given”
”osiping: Invalid address given”
”osiping: su mode required”
”osiping: OSI Stack is not licensed”
”osiping: OSI Stack is not Loaded”
”osiping: OSI Stack is Out Of Order”
”osiping: OSI Stack, no Subnet attached”
”osiping: TP1006 not present”

## Implementation Specifics

Part of the OSI Stack osi_frame package.

## File

| | |
|---|---|
| */usr/bin/osiping* | Command executable file. |

# Appendix A. Trace Report

The following chapters present the display format used for MAD trace.

This trace report is divided in 2 parts:

– OSI low layer trace report, on page A-1.
– OSI high layer trace report, on page A-7.

## OSI Low Layer Trace Report

This chapter describes the display format to show the lower layers and entities trace information. The following kernel entities are considered here:

COTP Access Method

COTP Driver

COTP Layer

CLNP Layer

X25.3 Access Method

X25.3 Layer

X25.2 Layer

ES–IS layer

LLC Layer

Timer

Administration

MAD

Note that the following explanations appear in the SMIT contextual help–on–line.

Trace items considered in this section are listed below:

### Kernel entity traces (internal trace)

All developers' traces concerning kernel entity processing.

The format of these traces is ASCII text.

These traces are not truncated.

The display format used for **Kernel entity traces** is the following:

***Entity_name* [Layer|Entity|Application]**

| **Time** | **Pid** | **Information** |
|---|---|---|
| hh:mm.ss.ms | *Pid* | *Info according to trace level* |

For example, the display format used for **MAD trace** is the following:

```
MAD Entity

Time            Pid     Information

15.07.33.066    SCHED0  MADMPX chan=C
                        MADOPEN dev=C rw=1
                        MADIOCTL dev=C cmd=1 ptarg=20051FC8
```

## ICB/UCB trace

| | |
|---|---|
| ICB | (Internal Control Block) is used to provide inter modules communication. |
| UCB | (User Control Block) is used to communicate between user application and OSI STACK. |

The display format used for **ICBs** is the following:

**CLNP Layer**

```
Time           Typ  Src   Dst   Function   Req  Size Lgi Nbi  Nbo  Data Error
13.21.15.546   ICB  CLNP  COTP  DATA       IND  036  4   0    1    18   0x00


ICB            |
               |(DATA) Standard data
               |(IND) Indication
               |
               |Address = 0x05c2e080
               |Source context = 0x0625af80
               |Destination context = 0x0623f400
               |Chain[01]= 0x0000065C  Offset= 0x000d  Length= 0x0012
               |
CLNP  COTP     | 4931 0004 0100 0400 0625 af80 0623 f400
               | 0410 0f10 0000 0000 0301 0600 0f00 0000
               | 0000 adbf 0000 065c 0012 000d
               |
>> chain 1 >>  | 0501 0203 0505 0401 0203 0400 0004 6f00
               | 9225
```

The display format used for **UCBs** is the following:

**USER Application**

```
Time           Typ  Src   Dst   Function   Req  Size Lgi Nbi  Nbo  Data Error
15.20.36.030   UCB  USER  ADMIN admACTION  REQ  036  8   0    0    0    0x00


UCB            |
               | (admACTION) Specific action on an object
               | (REQ) Request
               |
               | Address = 0x069d5450
               | Minor number = 12
               |
USER  ADMIN    | 5531 0008 0000 0000 0000 0000 2005 1fd0
               | 0010 000C 0000 0000 0300 01cb 0000 0000
               | 0110 0000 0000 0000
```

**ICB/UCB's – Summary Field Descriptions**

| | |
|---|---|
| **Time** | The time when the ICB/UCB was caught by MAD. |

| | |
|---|---|
| **Typ** | The type of control block. The possible values are: |
| | **ICB** Internal control block. |
| | **UCB** User control block. |

**Src and Dst**

Identifiers of the modules which send and receive ICB/UCBs.

Possible values are:

| | |
|---|---|
| ACSE | **A**ssociation **C**ontrol **S**ervice **E**lement |
| ADMIN | **ADMIN**istration |
| CLNP | **C**onnection**L**ess **N**etwork **P**rotocol |
| COPP | **C**onnection **O**riented **P**resentation **P**rotocol |
| COSP | **C**onnection **O**riented **S**ession **P**rotocol |
| COTP | **C**onnection **O**riented **T**ransport **P**rotocol |
| COTPam | **C**onnection **O**riented **T**ransport **P**rotocol **A**ccess **M**ethod |
| COTPdv | **C**onnection **O**riented **T**ransport **P**rotocol **D**ri**V**er |
| ES–IS | **E**nd **S**ystem – **I**ntermediate **S**ystem |
| GM | Memory Manager (**G**estionnaire **M**émoire) |
| LLC | **L**ogical **L**ink **C**ontrol |
| MAD | **M**éthode **d'A**ccès **D**istribuée |
| PX25 | **P**rocess **X25** |
| TIMER | **TIMER** |
| X25.2 | **X25** level **2** |
| X25.3 | **X25** level **3** |
| X253am | **X25** **A**ccess **M**ethod |
| USER | User application |
| ?<code> | Sub–system unknown |

**Function**

Code of the function service.

Possible values are:

**For the MAD:**

| | |
|---|---|
| ACTI | Specific action |
| ACON | Accept connection |
| ADD | Add attribute value |
| ALAM | IOP out of order |
| CONN | Connection |
| DATA | Standard data |
| DISC | Disconnection |
| EVT | Event |
| EXCE | Exception |
| EXPD | Expedited data |

| | |
|---|---|
| FOLD | Data to follow |
| INIS | Initialisation / Start–up |
| INIU | Initialisation / Updating |
| KILL | Client close |
| QALD | Qualified data |
| QFDA | Qualified and to follow |
| RCDA | Data with delivery confirmation |
| REIN | Re initialisation |
| TDAT | Data transfer |
| TECH | Bind |
| TIME | Timer expire |
| ?<code> | Function unknown |

**For Administration Access Method:**

| | |
|---|---|
| admACTION | Specific action on a object |
| admADD | Add object attribute |
| admCLOSE | Connection closing |
| admGET | Read object attribute |
| admGETEV | Read event in synchronous mode |
| admLIST | List object attributes |
| admOPEN | Connection opening |
| admOPENP | Passive open |
| admRMOVE | Remove object attribute |
| admSET | Update object attribute |
| admTSEV | Read event in asynchronous mode |
| ?<code> | Function unknown |

**For X25.3 Access Method:**

| | |
|---|---|
| max3AkOP | Connection confirm |
| max3CLOSE | Close a connection |
| max3OPEN | Initiate a connection |
| max3READ | Read an MAD signal |
| max3RESET | Re initialisation of a connection |
| max3RQfac | On–line facility registration |
| max3SD | Send data message |
| max3SDEX | Send expedited data message |
| max3TEST | Test an MAD signal |
| max3WaOP | Wait for a connection |
| ?<code> | Function not referenced |

**For COTP Access Method:**

| | |
|---|---|
| cotpACPT | Connection waiting |
| cotpCLOS | Connection closing |
| cotpOPENQ | Connection opening |
| cotpOPENR | Response to an open indication |
| cotpRDSI | Signal read |
| cotpSEND | Normal message sending |
| cotpSENX | Express message sending |
| cotpTSTSI | Signal test |
| ?<code> | Function unknown |

**Req**

Code of the request service.
Possible values are:

| | |
|---|---|
| CONF | Confirmation |
| IDRP | Indication with reply requested |
| IND | Indication |
| REP | Reply |
| REQ | Request |
| RQRP | Request with reply requested |
| ?<code> | Request unknown |

**Size**

Size of ICB/UCB header.

Decimal format, 3 digits.

**Lgi**

Size of internal data area.

Decimal format, 3 digits.

**Nbi**

For ICB's, always set to 0.

For UCB's, number of input descriptors.

Decimal format, 2 digits.

**Nbo**

For ICB's, number of chain identifiers.

For UCB's, number of output descriptors.

Decimal format, 2 digits.

**Data**

Length of data area.

Chains for ICB's, input/output descriptors for UCB's.

Decimal format, 5 digits.

**Error**

The error or diagnostic code.

Hexadecimal format, 8 digits.

**ICB/UCB's – Extended Field Descriptions**

**Address**

Address of ICB/UCB.

Hexadecimal format, 8 digits.

**Chain [i]**

ICB's only, Address of chain.

Hexadecimal format, 8 digits.

**Offset**

ICB's only, Offset of chain.

Hexadecimal format, 4 digits.

**Length**

ICB's only, Length of chain.

Hexadecimal format, 4 digits.

**Minor**

UCB's only, minor number.

Decimal format, 3 digits.

# OSI High Layer Trace Report

This chapter describes the display format to show the upper layers trace information. The following kernel entities are considered here:

ACSE layer

COPP layer

COSP layer

All developers' traces concerning OSI upper layers processing.

The format of these traces is given in a catalog file named ositrace.msg.

These traces can be truncated.

Note that the following explanations appear in the SMIT contextual help–on–line.

The common display format used for **OSI upper layers** is the following:

**Layer_name Layer**

| **Time** | **Pid** | **Connection Name** | **Connection Type** | **Trace Level** |
|---|---|---|---|---|
| hh:mm.ss.ms | *Pid* | *con_name* | *con_type* | *trace level* |

*Info according to trace level*

## Upper layers – Common Field Descriptions

***Layer_name***

Name of the OSI upper layer.

Possible values are :

| | |
|---|---|
| ACSE | Association Control Service Element |
| COPP | **C**onnection **O**riented **P**resentation **P**rotocol |
| COSP | **C**onnection **O**riented **S**ession **P**rotocol |
| ?<code> | Upper layer unknown |

**Pid**

Process identifier (or scheduler name).

**Connection Name**

Name of the connection.

Hexadecimal format, 8 digits.

**Connection Type**

Type of the connection.

Hexadecimal format, 8 digits.

**Trace Level**

Trace level.

Possible values are :

| | |
|---|---|
| ERR | System and protocol **ERR**ors |
| PDU SENT | **P**rotocol **D**ata **U**nits **SENT** by the layer |
| PDU REC | **P**rotocol **D**ata **U**nits **REC**eived by the layer |
| SDU SENT LOW | **S**ervice **D**ata **U**nits **SENT** by **LOW**er layer |

|  |  |
|---|---|
| SDU REC LOW | **S**ervice **D**ata **U**nits **REC**eived by LOWer layer |
| SDU SENT UPP | **S**ervice **D**ata **U**nits **SENT** by **UPP**er layer |
| SDU REC UPP | **S**ervice **D**ata **U**nits **REC**eived by **UPP**er layer |
| AUT | **T**ransitions in **AUT**omata |
| CONF | **CONF**iguration actions |
| MGT | **M**ana**G**emen**T** |
| FCT | Internal **F**un**CT**ions |
| FCT MORE | Internal **F**un**CT**ions (**MORE** information) |
| ENT PT | **ENT**ry **P**oin**T**s of the layer |
| AUT DBG | **AUT**omata **D**e**B**u**G** |
| STR MESS | Call to **STR**eams (**MESS**age sending) |
| STR QUEUE | Call to **STR**eams (**QUEUE** management) |
| STR MEM | Call to **STR**eams (**MEM**ory management) |
| STR MASK | Call to **STR**eams (**MASK** management) |
| SYS MEM | Call to **SYS**tem (**MEM**ory management) |
| SYS TIME | Call to **SYS**tem (**TIME**r management) |
| SYS LCK | Call to **SYS**tem (**L**o**CK** management) |
| PERF ENT PT | **PERF**ormance (**ENT**ry **P**oin**T**s of the layer) |
| PERF MESS | **PERF**ormance (stream **MESS**age handling) |
| PERF TIME HAND | **PERF**ormance (**TIME**r **HAND**lers) |
| LEVEL i | **OTHER** development trace level |

## Upper layers – Info according to trace level Description

**ERR** level: SYSTEM AND PROTOCOL ERRORS

**ACSE Layer**

```
Time          Pid      Connection Name   Connection Type   Trace Level
14.05.33.456  7426     0x00000004        0x00000007        ERR
```

*Error message given by the upper layer.*

**PDU SENT** level: PROTOCOL DATA UNITS SENT BY THE LAYER

The display format is the following:

**ACSE Layer**

```
Time          Pid      Connection Name   Connection Type   Trace Level
14.05.33.456  7426     0x00000004        0x00000007        PDU SENT
```

*Hexadecimal dump of the Protocol Data Unit.*

**PDU REC** level: PROTOCOL DATA UNITS RECEIVED BY THE LAYER

The display format is the following:

**ACSE Layer**

| **Time** | **Pid** | **Connection Name** | **Connection Type** | **Trace Level** |
|---|---|---|---|---|
| 14.05.33.456 | 7426 | 0x00000004 | 0x00000007 | **PDU REC** |

*Hexadecimal dump of the Protocol Data Unit.*

**SDU SENT LOW** level: SERVICE DATA UNITS SENT BY LOWER LAYER

The display format is the following:

**ACSE Layer**

| **Time** | **Pid** | **Connection Name** | **Connection Type** | **Trace Level** |
|---|---|---|---|---|
| 14.05.33.456 | 7426 | 0x00000004 | 0x00000007 | **SDU SENT LOW** |

*Hexadecimal dump of the buffer of the stream message.*

**SDU REC LOW** level: SERVICE DATA UNITS RECEIVED BY LOWER LAYER

The display format is the following:

**ACSE Layer**

| **Time** | **Pid** | **Connection Name** | **Connection Type** | **Trace Level** |
|---|---|---|---|---|
| 14.05.33.456 | 7426 | 0x00000004 | 0x00000007 | **SDU REC LOW** |

*Hexadecimal dump of the buffer of the stream message.*

**SDU SENT UPP** level: SERVICE DATA UNITS SENT BY UPPER LAYER

The display format is the following:

**ACSE Layer**

| **Time** | **Pid** | **Connection Name** | **Connection Type** | **Trace Level** |
|---|---|---|---|---|
| 14.05.33.456 | 7426 | 0x00000004 | 0x00000007 | **SDU SENT UPP** |

**ACSE Layer**

| **Time** | **Pid** | **Connection Name** | **Connection Type** | **Trace Level** |
|---|---|---|---|---|
| 14.05.33.456 | 7426 | 0x00000004 | 0x00000007 | **PDU SENT** |

*Hexadecimal dump of the buffer of the stream message.*

**SDU REC UPP** level: SERVICE DATA UNITS RECEIVED BY UPPER LAYER

The display format is the following:

**ACSE Layer**

| **Time** | **Pid** | **Connection Name** | **Connection Type** | **Trace Level** |
|---|---|---|---|---|
| 14.05.33.456 | 7426 | 0x00000004 | 0x00000007 | **SDU REC UPP** |

*Hexadecimal dump of the buffer of the stream message.*

**AUT** level : TRANSITIONS IN AUTOMATA

The display format is the following:

**ACSE Layer**

| **Time** | **Pid** | **Connection Name** | **Connection Type** | **Trace Level** |
|---|---|---|---|---|
| 14.05.33.456 | 7426 | 0x00000004 | 0x00000007 | **AUT** |

The following information is displayed:

**Automaton**

Automaton name given by the layer.

**Event**

Type of event receiving.
Hexadecimal format, 8 digits.

**Initial state**

Initial state before event receiving.
Hexadecimal format, 8 digits.

**Final state**

Final state after event receiving.
Hexadecimal format, 8 digits.

**ENT PT** level: ENTRY POINTS OF THE LAYER

The display format is the following:

**ACSE Layer**

| **Time** | **Pid** | **Connection Name** | **Connection Type** | **Trace Level** |
|---|---|---|---|---|
| 14.05.33.456 | 7426 | 0x00000004 | 0x00000007 | **ENT PT** |

The following information is displayed:

**Function**

Name of the function given by the layer.

**Param name**

Name of the parameter given by the layer.

**Param value**

Value of the parameter.
Format depends of the parameter type.

**FCT** level: INTERNAL FUNCTIONS

The display format is the following:

**ACSE Layer**

| **Time** | **Pid** | **Connection Name** | **Connection Type** | **Trace Level** |
|---|---|---|---|---|
| 14.05.33.456 | 7426 | 0x00000004 | 0x00000007 | **FCT** |

The following information is displayed:

**Function**

Name of the function given by the layer.

**Param name**

Name of the parameter given by the layer.

**Param value**

Value of the parameter.
Format depends of the parameter type.

**FCT MORE** level: INTERNAL FUNCTIONS (MORE INFORMATION)

The display format is the following:

**ACSE Layer**

| **Time** | **Pid** | **Connection Name** | **Connection Type** | **Trace Level** |
|---|---|---|---|---|
| 14.05.33.456 | 7426 | 0x00000004 | 0x00000007 | **FCT MORE** |

*Hexadecimal dump of buffer parameters.*

**AUT DBG** level: AUTOMATA DEBUG

The display format is the following:

**ACSE Layer**

| **Time** | **Pid** | **Connection Name** | **Connection Type** | **Trace Level** |
|---|---|---|---|---|
| 14.05.33.456 | 7426 | 0x00000004 | 0x00000007 | **AUT DBG** |

*Display given by the layer.*

**CONF** level: CONFIGURATIONS ACTIONS

The display format is the following:

**ACSE Layer**

| **Time** | **Pid** | **Connection Name** | **Connection Type** | **Trace Level** |
|---|---|---|---|---|
| 14.05.33.456 | 7426 | 0x00000004 | 0x00000007 | **CONF** |

*Message explaining the configuration treatment.*

**MGT** level: MANAGEMENT ACTIONS

The display format is the following:

**ACSE Layer**

| **Time** | **Pid** | **Connection Name** | **Connection Type** | **Trace Level** |
|---|---|---|---|---|
| 14.05.33.456 | 7426 | 0x00000004 | 0x00000007 | **MGT** |

*Message explaining the management treatment.*

**STR MESS** level: CALL TO STREAMS (MESSAGE SENDING)

The display format is the following:

**ACSE Layer**

| **Time** | **Pid** | **Connection Name** | **Connection Type** | **Trace Level** |
|---|---|---|---|---|
| 14.05.33.456 | 7426 | 0x00000004 | 0x00000007 | **STR MESS** |

The following information is displayed:

**Function**

Name of the stream function.
Possible values are:

| | |
|---|---|
| PUTNEXT | Put a message to the next queue |
| QREPLY | Send a message on a stream in the reverse direction |

**Queue**

Address of the stream queue.
Hexadecimal format, 8 digits

*Hexadecimal dump of the stream message*

**STR QUEUE** level: CALL TO STREAMS (QUEUE MANAGEMENT)

The display format is the following:

**ACSE Layer**

| **Time** | **Pid** | **Connection Name** | **Connection Type** | **Trace Level** |
|---|---|---|---|---|
| 14.05.33.456 | 7426 | 0x00000004 | 0x00000007 | **STR QUEUE** |

The following information is displayed:

Function

Name of the stream function.
Possible values are:

| | |
|---|---|
| ENABLEOK | Re–allow a queue to be scheduled for service |
| FLUSHQ | Flush a queue |
| GETQ | Get a message from a queue |
| INSQ | Put a message at a specific place in a queue |
| NOENABLE | Prevent a queue from being scheduled |
| PUTBQ | Return a message to the beginning of a queue |
| PUTQ | Put a message on a queue |
| QENABLE | Enable a queue |
| RMVQ | Remove a message from a queue |

**Queue**

Address of the stream queue.

**Message**

Address of the message.
Hexadecimal dump of the stream message.
Meaningful only for the following functions:

GETQ

INSQ

PUTBQ

PUTQ

RMVQ

**Flag**

Meaningful only for the FLUSHQ function.
Possible values are:

FLUSHDATA

FLUSHALL

*Hexadecimal dump of the stream message*

**STR MEM** level : CALL TO STREAMS (MEMORY MANAGEMENT)

The display format is the following:

**ACSE Layer**

| Time | Pid | Connection Name | Connection Type | Trace Level |
|---|---|---|---|---|
| 14.05.33.456 | 7426 | 0x00000004 | 0x00000007 | **STR MEM** |

The following information is displayed:

**Function**

Name of the stream function.
Possible values are:

| | |
|---|---|
| ALLOCB | Allocate a message and data block |
| BUFCALL | Recover from failure of ALLOCB |
| DUPB | Duplicate a message block descriptor |
| DUPMSG | Duplicate a message |
| ESBALLOC | Allocate message and data blocks |
| FREEB | Free a single message block |
| FREEMSG | Free all message blocks in a message |
| RMVB | Remove a message block from a message |
| UNBUFCALL | Cancel a BUFCALL request |

**Message**

Address of the message.
Hexadecimal format, 8 digits.
Meaningful only for the following functions:

DUPB

DUPMSG

ESBALLOC

FREEMSG

**Message Block**

Address of the message block.
Hexadecimal format, 8 digits.
Meaningful only for the following functions:

ALLOCB

FREEB

RMVB

**Size**

Size of the message.
Hexadecimal format, 8 digits.
Meaningful only for the following functions:

ALLOCB

BUFCALL

ESBALLOC

**Prio**

Priority of the request.
Possible values are:

LOW

MED

HIGH

Meaningful only for the following functions:

ALLOCB

BUFCALL

ESBALLOC

**Arg**

Argument of the BUFCALL function parameter.
Hexadecimal format, 8 digits.

**Id**

Identifier fo the UNBUFCALL request.
Hexadecimal format, 8 digits.

**Base**

Base for the ESBALLOC request.
Hexadecimal format, 8 digits.

**STR MASK** level: CALL TO STREAMS (MASK MANAGEMENT)

The display format is the following:

**ACSE Layer**

| **Time** | **Pid** | **Connection Name** | **Connection Type** | **Trace Level** |
|---|---|---|---|---|
| 14:05.33.456 | 7426 | 0x00000004 | 0x00000007 | **STR MASK** |

The following information is displayed:

**Function**

Name of the stream function.
Possible values are:

| | |
|---|---|
| MASK | Mask stream at level. |
| DMASK | Restore interrupt level. |

**ITlevel**

Level of interrupt.

**SYS MEM** level: CALL TO SYSTEM (MEMORY MANAGEMENT)

The display format is the following:

**ACSE Layer**

| **Time** | **Pid** | **Connection Name** | **Connection Type** | **Trace Level** |
|---|---|---|---|---|
| 14.05.33.456 | 7426 | 0x00000004 | 0x00000007 | **SYS MEM** |

The following information is displayed:

**Function**

Name of the system function.
Possible values are:

| | |
|---|---|
| ALLOCBUF | Allocate a buffer |
| FREEBUF | Release a buffer |
| ALLOCCTX | Allocate a context |
| FREECTX | Release a context |

**Buffer**

Address of the buffer.
Hexadecimal format, 8 digits.

**Size**

Size of the buffer.
Hexadecimal format, 8 digits.

**SYS TIME** level: CALL TO SYSTEM (TIMER MANAGEMENT)

The display format is the following:

**ACSE Layer**

| **Time** | **Pid** | **Connection Name** | **Connection Type** | **Trace Level** |
|---|---|---|---|---|
| 14.05.33.456 | 7426 | 0x00000004 | 0x00000007 | **SYS TIME** |

The following information is displayed:

**Function**

Name of the system function.
Possible values are:

| | |
|---|---|
| ACT | Timer activation. |
| DACT | Timer deactivation. |
| MASK | Mask timer at level. |
| DMASK | Restore interrupt level. |

**ITlevel**

Level of interrupt.

**SYS LOCK** level: CALL TO SYSTEM (LOCK MANAGEMENT)

The display format is the following:

**ACSE Layer**

| **Time** | **Pid** | **Connection Name** | **Connection Type** | **Trace Level** |
|---|---|---|---|---|
| 14.05.33.456 | 7426 | 0x00000004 | 0x00000007 | **SYS LOCK** |

The following information is displayed:

**Function**

Name of the critical function given by the upper layer.
Possible values are:

| | |
|---|---|
| ENTER | Entering in critical section of code. |
| EXIT | Exiting from critical section of code. |

**Lock**

Address of the lock.
Hexadecimal format, 8 digits.

**PERF ENT PT** level: PERFORMANCE (ENTRY POINTS OF THE LAYER)

The display format is the following:

**ACSE Layer**

| **Time** | **Pid** | **Connection Name** | **Connection Type** | **Trace Level** |
|---|---|---|---|---|
| 14.05.33.456 | 7426 | 0x00000004 | 0x00000007 | **PERF ENT PT** |

**ACSE Layer**

| **Time** | **Pid** | **Connection Name** | **Connection Type** | **Trace Level** |
|---|---|---|---|---|
| 14:05.33.456 | 7426 | 0x00000004 | 0x00000007 | **PERF TIME PT** |

*Information given by the layer, according to this level.*

**PERF MESS** level: PERFORMANCE (STREAM MESSAGE HANDLING)

The display format is the following:

**ACSE Layer**

| **Time** | **Pid** | **Connection Name** | **Connection Type** | **Trace Level** |
|---|---|---|---|---|
| 14.05.33.456 | 7426 | 0x00000004 | 0x00000007 | **PERF MESS** |

**ACSE Layer**

| **Time** | **Pid** | **Connection Name** | **Connection Type** | **Trace Level** |
|---|---|---|---|---|
| 14.05.33.456 | 7426 | 0x00000004 | 0x00000007 | **AUT DBG** |

*Information given by the layer, according to this level.*

**PERF TIME HAND** level: PERFORMANCE (TIMER HANDLERS)

The display format is the following:

**ACSE Layer**

| **Time** | **Pid** | **Connection Name** | **Connection Type** | **Trace Level** |
|---|---|---|---|---|
| 14.05.33.456 | 7426 | 0x00000004 | 0x00000007 | **PERF TIME HAND** |

*Information given by the layer, according to this level.*

**OTHER** levels: OTHER DEVELOPMENT TRACE LEVELS

The display format is the following:

**ACSE Layer**

| **Time** | **Pid** | **Connection Name** | **Connection Type** | **Trace Level** |
|---|---|---|---|---|
| 14.05.33.456 | 7426 | 0x00000004 | 0x00000007 | **LEVEL i** |

*Info according to other development trace level.*

# Bibliography

## List of Applications Documentation

### XAP

*ACSE/P to XAP Interface Porting Guide* (86 A2 46AP)

*XAP API Reference Manual* (86 A2 27AP)

*ACSE/Presentation Services API (XAP) CAE Specification* (86 A2 30AP)

### XTI

*XTI/XX25 Administrator and User Guide* (86 A2 04AP)

*XTI Porting Guide* (86 A2 25AP)

*X/Open Transport Interface XPG4 CAE Specification Version 2* (40 A2 49AS)

### OSI API

*Session Access Library Programmer's Guide* (86 A2 48WE)

*ROSE Interface Library Programmer's Guide* (86 A2 31AP)

## List of Related Documents

- Bull DPX/20 *OSI Services Reference Manual* (86 A2 05AQ)
- Bull DPX/20 *OSI Communications Porting Guide* (86 A2 44AP)

## Standards

### ISO

| | |
|---|---|
| ISO 8073 | Connection–oriented transport protocol definition |
| ISO 8327 | Connection–oriented session protocol definition |
| ISO 8650 | Protocol specification for the Association Control Service Element (ACSE) |
| ISO 8473 | Protocol for providing the connectionless–mode network service |
| ISO 9542 | End System to Intermediate System routing exchange protocol for use in conjunction with ISO 8473 |
| ISO 8208 | X.25 Packet layer protocol for Data Terminal Equipment |

### IEEE

| | |
|---|---|
| IEEE 8802.2 | Local Area Networks – Logical Link Control (LLC) |

## License Control

License control details are provided in the following documents:

*AIX iFOR/LS System Management Guide* (86 A2 11AQ)

*AIX iFOR/LS Tips and Techniques* (86 A2 12AQ)

# Glossary

Listed here are the abbreviations, key–words and phrases used in this document.

**ACSE**
Association Control Service Element

**ASCII**
American national Standard Code for Information Interchange

**CLNP**
ConnectionLess Network Protocol

**COSP**
Connection Oriented Session Protocol

**COTP**
Connection Oriented Transport Protocol

**DCO**
Distributed Computing Operation

**ES–IS**
End System, Intermediate System

**FDDI**
Fibre Distributed Data Interface

**FIFO**
First In First Out

**ICB**
Internal Control Block
Used for inter sub–system communication

**LAN**
Local Area Network

**LLC**
Logical Link Control
Top sub–layer in the data link layer. Used to facilitate connection with LAN (Local Area Network)

**LSAP**
Link Service Access Point
Access points to LLC services

**MAC**
Media Access Control
Low sub–layer in the data link layer. Used to facilitate connection with LAN

**MAD**
Distributed Access Method
Uses a set of primitives to define relations between client processes and server processes which manage the communications services

**NPDU**
Network Protocol Data Unit

**NSDU**
Network Service Data Unit

**NSAP**
Network Service Access Point
Access points to Network services

**ODIT**
OSI diagnostic Interactive Tool

**ODM**
Object Database Management

**OSI**
Open Systems Interconnection

**PDU**
Protocol Data Unit

**PICS**
Protocol Implementation Conformance Statements

**SAP**
Service Access Point

**SMIT**
System Management Interactive Tool

**SPDU**
Session Protocol Data Unit

**SSAP**
Session Service Access Point
Access points to Session services

**TPI**
Transport Protocol Interface

**TPDU**
Transport Protocol Data Unit

**TSAP**
Transport Service Access Point
Access points to Transport services

**UCB**
User Control Block
Used for communication between user processes and OSI driver

**WAN**
Wide Area Network

**X.25**
Network protocol for WAN

**XTI**
X/Open Transport Interface

**Xwindow**
Graphic user interface

# Index

# Vos remarques sur ce document / Technical publication remark form

| | |
|---|---|
| **Titre / Title :** | Bull DPX/20 OSI Diagnostic Interactive Toolkit (ODIT) |

| | |
|---|---|
| **Nº Reférence / Reference Nº :** 86 A2 39WG 03 | **Daté / Dated :** June 1996 |

ERREURS DETECTEES / ERRORS IN PUBLICATION

AMELIORATIONS SUGGEREES / SUGGESTIONS FOR IMPROVEMENT TO PUBLICATION

Vos remarques et suggestions seront examinées attentivement
Si vous désirez une réponse écrite, veuillez indiquer ci-après votre adresse postale complète.

Your comments will be promptly investigated by qualified technical personnel and action will be taken as required.
If you require a written reply, please furnish your complete mailing address below.

NOM / NAME : ____________________ Date : __________

SOCIETE / COMPANY : ____________________

ADRESSE / ADDRESS : ____________________

____________________

Remettez cet imprimé à un responsable BULL ou envoyez-le directement à :

Please give this technical publication remark form to your BULL representative or mail to:

**Bull Electronics Angers S.A.**
**CEDOC**
Atelier de Reprographie
331 Avenue Patton
49004 ANGERS CEDEX 01
FRANCE

PLACE BAR CODE IN LOWER
LEFT CORNER

**Bull Electronics Angers S.A.**
**CEDOC**
Atelier de Reprographie
331 Avenue Patton
49 004 ANGERS CEDEX 01
FRANCE

ORDER REFERENCE
**86 A2 39WG 03**

Utiliser les marques de découpe pour obtenir les étiquettes.
Use the cut marks to get the labels.

**DPX/20**
AIX
OSI Diagnostic
Interactive Toolkit
(ODIT)

86 A2 39WG 03

**DPX/20**
AIX
OSI Diagnostic
Interactive Toolkit
(ODIT)

86 A2 39WG 03

**DPX/20**
AIX
OSI Diagnostic
Interactive Toolkit
(ODIT)

86 A2 39WG 03