Earth Observing System Data and Information System (EOSDIS) Test System (ETS) Detailed Design Specification Volume 1 Index and Support Software

March 1996

Prepared Under Contract NAS5-31500
Task Assignment 15 507
By Computer Sciences Corporation
515-DDS-001

Prepared by:

J. McGarry Date
Task Leader, CSC

Concurred by:

W. Fuller Date
ETS Project Manager, GSFC, Code 515.1

Reviewed by:

M. Heffernan Date
Manager, Simulation Systems Department, CSC

Concurred by:

D. Lakins Date
ESDIS ETS Manager, GSFC, Code 505

Test Systems Branch
Goddard Space Flight Center
Greenbelt, Maryland

Preface

This document is volume 1 of a 5 volume set specifying the detailed design for the Earth Observing System Data and Information System (EOSDIS) Test System (ETS). This volume describes the Control and Display Subsystem (CDS) software exclusive of the Telemetry Processing Control Environment (TPCE) and the Simulated CCSDS Telemetry Generator (SCTGEN), software being developed for all three ETS systems, as well as providing an index to the other ETS volumes. TPCE and SCTGEN are NASA Code 520 products that are described in other volumes in the set.

Abstract

The Earth Observing System Data and Information System (EOSDIS) Test System (ETS) provides test capabilities for the ground system supporting Earth-observing spacecraft that conform to the Consultative Committee for Space Data Systems (CCSDS) Recommendations. ETS is designed to simulate and test EOSDIS ground systems and element interfaces. The simulation and testing capabilities cover only level-zero processing, including data delivery to the Distributed Active Archive Centers (DAACs).

This document provides the detailed design specifications for the ETS support software being developed for the Control and Display Subsystem.

Keywords: *Earth Observing System Data and Information System (EOSDIS) Test System, ETS, simulation, design*

Contents

Preface

Abstract

Section 1. Introduction

1.1 Purpose and Scope 1-1

1.2 ETS Document Set 1-1

1.3 Document Organization 1-2

1.4 Applicable Documents 1-3

1.5 Definitions and Conventions 1-3

Section 2. ETS Design Overview

2.1 Background 2-1

2.2 ETS Test Configurations 2-1

2.3 ETS Architecture 2-1

2.4 Software Configuration Item Overview 2-3
2.4.1 ETS Low-Rate System 2-3
2.4.2 ETS High-Rate System 2-5
2.4.3 ETS Multimode Portable Simulator 2-5

2.5 Design Methodology 2-6
2.5.1 Requirements Definition 2-6
2.5.2 Design 2-6
2.5.3 Implementation 2-9
2.5.4 Qualification Testing 2-9

2.6 Design Considerations 2-9
2.6.1 Design Drivers 2-9
2.6.2 Design Modifications since the ETS Design Review 2-10

2.7 Development Environment 2-10

Section 3. Software Configuration Item Detailed Design Descriptions

3.1 Main Menu Controller 3-1
3.1.1 Description 3-1
3.1.2 User Interface Design 3-1
3.1.3 Processing 3-2
3.1.4 Prologs/Program Design Language/Structure Charts for Module Drivers 3-2

3.2 OMD Simulator 3-4
3.2.1 Description 3-4
3.2.2 User Interface Design 3-6
3.2.2.1 Definitions and Conventions 3-6
3.2.2.2 OMD Simulator Main Window 3-6
3.2.2.3 Scenario Execution 3-8
3.2.2.4 Transmission Test 3-9

3.2.2.5 Main Window Menu Bar 3-9
3.2.2.5.1 File 3-10
3.2.2.5.2 View 3-12
3.2.2.6 Scenario Editor 3-12
3.2.2.7 Scenario Editor Menu Bar 3-13
3.2.2.7.1 File 3-13
3.2.2.7.2 Edit 3-14
3.2.2.8 OMD Editor 3-15
3.2.2.9 OMD Editor Menu Bar 3-16
3.2.2.9.1 File 3-16
3.2.2.9.2 Edit 3-17
3.2.2.10 CODA Report Editor Panels 3-17
3.2.2.10.1 CODA Report Content Indicators 3-18
3.2.2.10.2 Return Link Processing Function 3-19
3.2.2.10.3 Forward Link Processing Function 3-20
3.2.2.10.4 Return Link Service Status 3-21
3.2.2.10.5 Forward Link Service Status 3-22
3.2.2.10.6 Ground Communications Return Link 3-22
3.2.2.10.7 Accounting Information Received 3-23
3.2.2.10.8 Accounting Information Transmitted 3-23
3.2.2.11 EDS/PDS Delivery Request Editor Panels 3-24
3.2.2.12 Ground Message Header 3-27
3.2.2.13 OMD Traffic 3-28
3.2.3 Processing 3-28
3.2.4 Prologs/Program Design Language/Structure Charts for Module Drivers 3-31
3.2.4.1 om_scenario_manager 3-31
3.2.4.2 om_scheduler 3-31
3.2.4.3 om_transmission_task 3-37
3.2.4.4 om_logger 3-41
3.2.4.5 om_reception_task 3-44
3.2.4.6 om_delogger 3-47

3.3 Rate Buffered File Utility 3-49
3.3.1 Description 3-49
3.3.2 User Interface Design 3-49
3.3.3 Processing 3-50
3.3.4 Prologs/Program Design Language/Structure Charts for Module Drivers 3-50

3.4 Command Archive Utility 3-52
3.4.1 Description 3-52
3.4.2 User Interface Design 3-52
3.4.3 Processing 3-53
3.4.4 Prologs/Program Design Language/Structure Charts for Module Drivers 3-53

3.5 Data Verification Tool 3-55
3.5.1 Description 3-55
3.5.2 User Interface Design 3-55
3.5.2.1 Definitions and Conventions 3-55
3.5.2.2 DVT Main Window 3-55
3.5.2.3 DVT Main Window Menu Bar 3-57
3.5.2.3.1 File 3-58
3.5.2.3.2 View 3-59
3.5.2.3.3 Config 3-59
3.5.2.4 Default Parameters Window 3-59
3.5.3 Processing 3-59

3.5.4 Prologs/Program Design Language/Structure Charts for Module Drivers 3-60
3.5.4.1 dv_main 3-64
3.5.4.2 dv_verify_cr 3-65
3.5.4.3 dv_verify_pkthdr 3-66
3.5.4.4 dv_create_report 3-68

3.6 Data Comparison Tool 3-69
3.6.1 Description 3-69
3.6.2 User Interface Design 3-70
3.6.2.1 Definitions and Conventions 3-70
3.6.2.2 DCT Main Window 3-70
3.6.2.3 DCT Main Window Menu Bar 3-71
3.6.2.3.1 File 3-72
3.6.2.3.2 View 3-72
3.6.3 Processing 3-72
3.6.4 Prologs/Program Design Language/Structure Charts for Module Drivers 3-72
3.6.4.1 dc_main 3-76
3.6.4.2 dc_compare_cr 3-77
3.6.4.3 dc_compare_pkthdr 3-78
3.6.4.4 dc_create_report 3-80

Section 4. Size Estimates

Appendix A. Support Software Requirements to Design Traceability

Appendix B. Data Dictionary

Appendix C. Abbreviations and Acronyms

List of Figures

Figure 2-1. ETS Architecture 2-2
Figure 2-2. System CI Architecture 2-4
Figure 2-3. Structure Chart Conventions 2-8
Figure 3-1. Main Window for ETS LRS Menu Controller 3-2
Figure 3-2. Structure Chart for Menu Controller 3-3
Figure 3-3. OMD Simulator Main Window 3-7
Figure 3-4. Main Window Menu Bar 3-10
Figure 3-5. Scenario Editor 3-12
Figure 3-6. Scenario Editor Menu Bar 3-13
Figure 3-7. Scenario Editor Confirmation Boxes 3-14
Figure 3-8. OMD Editor 3-15
Figure 3-9. OMD Editor Menu Bar 3-16
Figure 3-10. OMD Editor Question Boxes 3-17
Figure 3-11. CODA Report Content Indicators 3-18
Figure 3-12. Return Link Processing Function 3-19
Figure 3-13. Forward Link Processing Function 3-20
Figure 3-14. Return Link Service Status 3-21
Figure 3-15. Ground Communications Return Link 3-22
Figure 3-16. Accounting Information Received 3-23
Figure 3-17. Accounting Information Transmitted 3-23
Figure 3-18. Exchange Data Unit Label 3-24
Figure 3-19. PDS/EDS Delivery Record Lable 3-25
Figure 3-20. PDS/EDS Delivery Record PVL Statements 3-26

Figure 3-21. Ground Message Header Dialog Box 3-27
Figure 3-22. OMD Traffic Dialog Box 3-28
Figure 3-23. Structure Chart for OM_Main 3-30
Figure 3-24. Structure Chart for OM_Scenario_Manager 3-33
Figure 3-25. Structure Chart for OM_Scheduler 3-36
Figure 3-26. Structure Chart for OM_Transmission_Task 3-40
Figure 3-27. Structure Chart for OM_Logger 3-43
Figure 3-28. Structure Chart for OM_Reception_Task 3-46
Figure 3-29. Structure Chart for OM_Delogger 3-48
Figure 3-30. Main Window for Rate Buffering File Utility 3-49
Figure 3-31. Structure Chart for Rate Buffering Process 3-51
Figure 3-32. Main Window for Command Archive Utility 3-52
Figure 3-33. View Window for Command Archive Files 3-53
Figure 3-34. Structure Chart for Command Archive Process 3-54
Figure 3-35. Main Window for Data Verification Tool 3-56
Figure 3-36. DVT Main Window Menu Bar 3-58
Figure 3-37. Structure Chart for Data Verification Tool (dv_main_task) 3-61
Figure 3-38. Structure Chart for Data Verification Tool (dv_verify_cr) 3-62
Figure 3-39. Structure Chart for Data Verification Tool (dv_verify_pkthdr) 3-63
Figure 3-40. Main Window for Data Comparison Tool 3-70
Figure 3-41. DCT Main Window Menu Bar 3-71
Figure 3-42. Structure Chart for Data Comparison Tool (dc_main_task) 3-73
Figure 3-43. Structure Chart for Data Comparison Tool (dc_compare_cr) 3-74
Figure 3-44. Structure Chart for Data Comparison Tool (dc_compare_pkthdr) 3-75

List of Tables

Table 1-1. Summary of ETS Detailed Design Documents 1-2
Table 3-1. Menu Controller Software Routines 3-4
Table 3-2. OMDSIM Software Routines 3-29
Table 4-1. Support Software Code Estimates 4-1

Section 1. Introduction

This document is volume 1 of a 5 volume set specifying the detailed design specifications for the Earth Observing System Data and Information System (EOSDIS) Test System (ETS) This volume describes the design of the Control and Display Subsystem software exclusive of the Telemetry Control Environment (TPCE) and the Simulated CCSDS Telemetry Generator (SCTGEN). A guide to the other ETS design documents is provided in section 1.2.

1.1 Purpose and Scope

The ETS Control and Display Subsystem provides the operator interface for configuring and monitoring the Low-Rate and High-Rate Subsystems. The CDS also provides utilities for the generation and analysis of test data. The configuration items (CIs) discussed in this volume are the Main Menu Controller, the OMD Simulator, and various data manipulation utilities.

The Main Menu Controller makes up the highest level of user system interface. Its primary function is the integration of the various reused, government off-the-shelf (GOTS) and new software products.

The Operations Management Data Simulator (OMDSIM) provides a source of OMD messages to supplement ETS command and telemetry data streams. OMDSIM can also provide automated responses to received OMDs.

Data manipulation utilities currently include the Rate Buffered File utility, the Command Archive utility, the Data Verification Tool (DVT), and the Data Comparison Tool (DCT).

1.2 ETS Document Set

Just as ETS is a multipurpose test tool made up of GOTS products, new, and reused components, the ETS detailed design specifications are made up of documents from different sources. The document set is summarized in Table 1-1.

Volume 1 describes support software for the Control and Display Subsystem. This volume describes software to be written and integrated by CSC for NASA Code 515. Being the first document in the set, it provides an overview of ETS, and provides a road map to the other documents.

Volume 2 describes the design for the ETS Multimode Portable Simulator (MPS). Since the MPS is largely reused from the X-Ray Timing Explorer (XTE) Test and Training Simulator (XTTS), which was developed by CSC for NASA Code 515, that system's design document's format was also reused.

Volume 3 describes the ETS High-Rate System (HRS). This volume describes the HRS very large scale integration (VLSI) components, the Tape Recording Subsystem (TRS) and the user interface to control the VHS and TRS, which is based on TPCE. This is a document from the Microeletronic Systems Branch (MSB).

Volume 4 describes the ETS Low-Rate System (LRS). This volume describes the LRS very large scale integration (VLSI) components and the user interface to control them, which is based on TPCE. This document is also provided by the MSB.

Volume 5 describes the design of the Simulated CCSDS Telemetry Generator (SCTGEN). This product is being developed by the MSB.

Table 1-1. Summary of ETS Detailed Design Documents

Vol. No.	Description	Component	Author
1	ETS Index and Support Software Detailed Design Specification	CDS	CSC/515
2	ETS MPS Detailed Design Specification	MPS	CSC/515
3	ETS HRS Detailed Design Specification	HRS	NASA/521
4	ETS LRS Detailed Design Specification	LRS	NASA/521
5	SCTGEN Detailed Design Specification	SCTGEN	NASA/521

1.3 Document Organization

This document, Volume 1 of the ETS detailed design specifications, contains four sections and three appendices. Section 1 provides an introduction to this document and describes the entire document set. This section provides Volume 1 document organization, applicable documents, and definitions and conventions.

Section 2 provides an overview of the ETS system design and the methodology used for the system support software and OMDSIM. This section also illustrates the decomposition of ETS into its major configuration items (CIs) and maps the CIs to the documents in the ETS detailed design specifications set.

Section 3 contains the detailed design (i.e., structure charts, prologs, and program design language) for the CDS support software. It contains separate subsections for the ETS Main Menu Controller, OMD Simulator, Rate Buffered Utility, Command Archive Utility, Data Verification Utility and the Data Comparison Utility.

Section 4 contains size estimates for the support software.

Appendix A contains a traceability matrix of the level 4 requirements mapped to the design elements contained in this volume. Each volume of this document includes a traceability matrix that maps the level 4 requirements to the design it contains.

Appendix B contains the high level data items for the structure charts, prologs and program design language (PDL). (Not all the data shown on structure charts and prologs is shown.)

Appendix C contains a list of abbreviations and acronyms.

1.4 Applicable Documents

1. National Aeronautics and Space Administration (NASA), Goddard Space Flight Center (GSFC), 515-4FRD/0294 (CSC/TR-94/6084), *Earth Observing System (EOS) Data and Information System (EOSDIS) Test System (ETS) Functional and Performance Requirements*, October 1995

2. --, 515-3OCD0194 (CSC/TR-94/6085), *Earth Observing System Data and Information System (EOSDIS) Test System (ETS) Operations Concept*, May 1995

3. --, 515-4SDD/0195 (CSC/SD-95/6025), *Earth Observing System Data and Information System (EOSDIS) Test System (ETS) System Design Specification*, May, 1995

4. --, (CSC/515-SRS-001), *Earth Observing System (EOS) Data and Information System (EOSDIS) Test System (ETS) Requirements Specification, Volume 1 Index and Support Software*, October 1995

5. --, DSTL-92-007, *Human-Computer Interface Guidelines*, August 1992

6. NASA Automated Information Security Handbook NHB 2410, 9A, June 1993

7. --, *NASA Communications (Nascom) Access Protection Policy and Guidelines*, Revision 3, August 1995

8. Open Software Foundation (OSF), *OSF/Motif Style Guide*, Revision 1.2, 1993

9. Consultative Committee for Space Data Systems (CCSDS), CCSDS 301.0-B-2, Recommendation for Space Data System Standards. Time Code, Blue Book, Issue-2, April 1990

10. TRW, 510-ICD-EDOS/EGS, (TRW 2311 B301.00), *Interface Control Document between the Earth Observing System (EOS) Data and Operations System (EDOS) and the EOS Ground System (EGS)*, January 1996

1.5 Definitions and Conventions

The graphical user interfaces (GUI) for the ETS systems described in Volume 1 and Volume 2 of this document adhere to the following conventions.

The widgets are operated and behave as described in the OSF Motif Style Guide. Features not described there, such as combination boxes are operated and behave as in MS Word 6.0a. (A combination box consists of a label, an editable field, a down arrow button, and a popup menu. The font selector on the MS Word tool bar is a combination box.).

Within this document:

Options are in italic.

Button names are bold.

In lists, the **selected** item is typed in bold letters and has a gray background.

A solid border around a labeled box indicates the field can be edited.

A dashed border indicates the field is for display only.

Within the overall design of OMDSIM, Support Utilities and MPS:

Most dialog boxes have an **OK** button, a **Cancel** button and a **Help** button. Selecting **OK** causes any input to be accepted, the task to be executed, and closes the dialog box. Selecting **Cancel** only closes the dialog box; no input is accepted and no task is executed. Selecting **Help** causes information about the dialog box to be displayed.

Dialog boxes that are used for display have **Close** buttons instead of **OK** and **Cancel**. The **Close** button closes the dialog box and no task is executed.

Some tasks require a confirmation before being started. Confirmation dialog boxes are brought up by selecting **OK** for these tasks. These dialog boxes have **Yes** and **No** buttons. Selecting **Yes**, continues with the **OK** callback function and closes both the confirmation dialog box and the task dialog box. Selecting **No** closes the confirmation dialog box but leaves the task dialog box in place; no task is executed.

Dialog boxes that don't have **Help** buttons are brought up by some action on a task dialog box. Help information for these dialog boxes is included in the help information in the task dialog box.

Some dialog boxes allow for modifications of many of the same type of item. Where practical, all items are displayed at the same time and each is edited and then set or reset with the **Set** or **Reset** buttons in the section above it. If there are too many items possible or they contain too much information to be displayed in a simple way, only one is displayed at a time. These sections have a **Set** button, a **Reset** button and a **Next** button. Selecting **Set** causes the parameters for that item to be saved for later use. Selecting **Reset** restores the parameters that were previously saved for that item. Selecting **Next** sets all parameters for the current item and displays the next item in the list.

The title of a dialog box identifies the action and the specific object of the task being performed.

One character, the mnemonic, of <u>M</u>enu names and some menu options is underlined. This allows the operator to access these items via accelerators. Accelerators are key bindings that can be used to activate an item without the mouse.

An ellipsis (...) on a menu option indicates a dialog box is popped up to complete the task.

A greater than sign (>) on a menu option indicates a sub-menu (cascade menu) appears for further qualification.

A rectangle to the right of a button indicates an option menu. Selecting the button causes a list of options to appear. Selecting an item in the list selects the current option. The current option is always displayed.

A square to the left of a label or menu option indicates that the item is a toggled option. When there are more than one in a panel or menu, more than one can be selected at a time. These are often called check boxes. The square is darkened when the option is set.

A diamond to the left of a label or menu option also indicates that the item is a toggled option. However, these options appear in groups of 2 or more and only one option can be selected at a time. These are often called radio buttons. The diamond appears darkened when the option is set.

Section 2. ETS Design Overview

2.1 Background

The requirements for ETS have evolved throughout several years as the EOSDIS ground system has undergone significant changes and the need for test and simulation capabilities has become better defined. The current ETS operations concept provides for a system which will support a set of ESDIS test configurations. The ETS operations concept uses three separate components from two fixed locations, including a portable component for contingency network interface testing. More background information is contained in the ETS Operations Concept (Reference 2) and the ETS System Design (Reference 3).

2.2 ETS Test Configurations

The ESDIS Project identified five test configurations in which ETS would provide test and simulation capabilities in support of EOSDIS testing. The capabilities provided by the five test configurations are:

1. Provide high-rate spacecraft data at the TDRSS Ground Terminal (TGT) interface to EOS Data and Operations System (EDOS)
2. Simulate and transmit EDOS data sets to a Distributed Active Archive Center (DAAC)
3. Simulate the front-end of a DAAC and receive EDOS data sets
4. Provide low-rate spacecraft simulations for two circumstances:
    - 4A—Support EOSDIS normal and contingency-mode operations when EDOS is available for forward- and return-link processing
    - 4B—Support tests with the EOS Operations Center (EOC) when EDOS is not available
5. Provide low-rate EDOS functions for EOC to interface with the EOS AM-1 spacecraft integration and test facility (SCITF) and the EOS AM-1 spacecraft simulator (SSIM)

2.3 ETS Architecture

Figure 2-1 shows the ETS architecture. ETS is a collection of three simulators: a Low-Rate System (LRS), a High-Rate System (HRS), and a Multimode Portable Simulator (MPS). The ETS LRS provides a functional EDOS interface between the EOC and the SCITF or SSIM for test and training purposes. The ETS HRS supports tests for EOSDIS return-link science data processing and interface functions. The ETS MPS supports testing for forward-link and non-science return-link processing and provides dynamic spacecraft telemetry simulations and command validation.

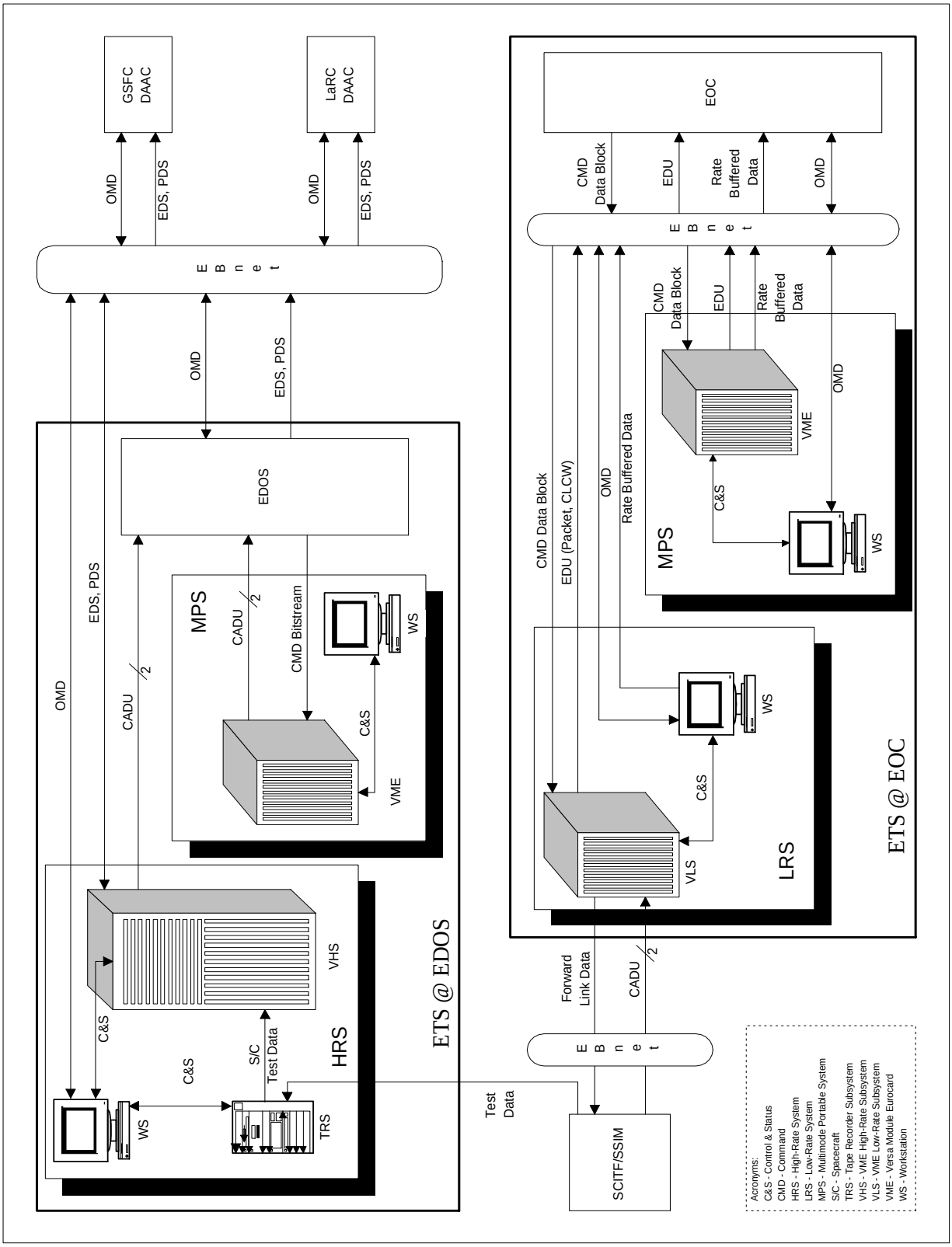

Figure 2-1. ETS Architecture

2.4 Software Configuration Item Overview

The three systems have been divided into subsystems and hardware and software CIs, as shown in Figure 2-2. The Control and Display Subsystem (CDS) is common to the HRS and to the LRS. The software specifications contained in this document are part of the CDS. Some of the functionality provided by the LRS is also common to the MPS.

2.4.1 ETS Low-Rate System

The LRS provides a functional EDOS interface for command and telemetry data flows between the EOC and SCITF or SSIM. The LRS provides the following functions:

- Receive low-rate spacecraft data as channel access data units (CADUs) in serial bitstream from the SCITF and the SSIM, perform frame synchronization, Reed-Solomon decoding and error correction, packet reassembly, command link control word (CLCW) extraction, and EDOS data unit (EDU) construction
- Annotate packet (in appended EDOS service header) with actual quality and accounting information (rather than simulated header fields)
- Transmit EDUs (packets and CLCWs) in real time to the EOC
- Provide rate buffering of low-rate playback data by creating a file of EDUs and performing a file transfer of the rate-buffered data file to the EOC
- Support two return link channels simultaneously
- Receive command data blocks from the EOC
- Perform EDOS forward-link processing, i.e., verify ground message headers, extract command link transmission units (CLTUs) and acquisition sequences, and output as serial bitstream
- Transmit forward link as serial bitstream (clock and data) to SCITF or SSIM
- Generate and transmit customer operations data accounting (CODA) reports in real time to reflect the actual return- and forward-link data processing

The LRS is composed of the VME low-rate subsystem (VLS) and the CDS. Most of the forward- and return-link processing functions listed above are provided by the VLS. The CDS supplies the user interface, provides real-time control, and is used to monitor operational status, functions provided by TPCE. In addition, the CDS support software provides the rate buffered file production and CODA generation functions listed above as part of the Utilities and OMD CIs.

The LRS detailed design specifications, excluding those of the CDS support software CIs, are contained in Volume 4. The LRS CDS support software design specifications are contained in this volume. Section 3.1 contains the Main Menu Controller design. Section 3.2 describes the OMDSIM software. The rest of Section 3 describes the design of the other supporting utilities.

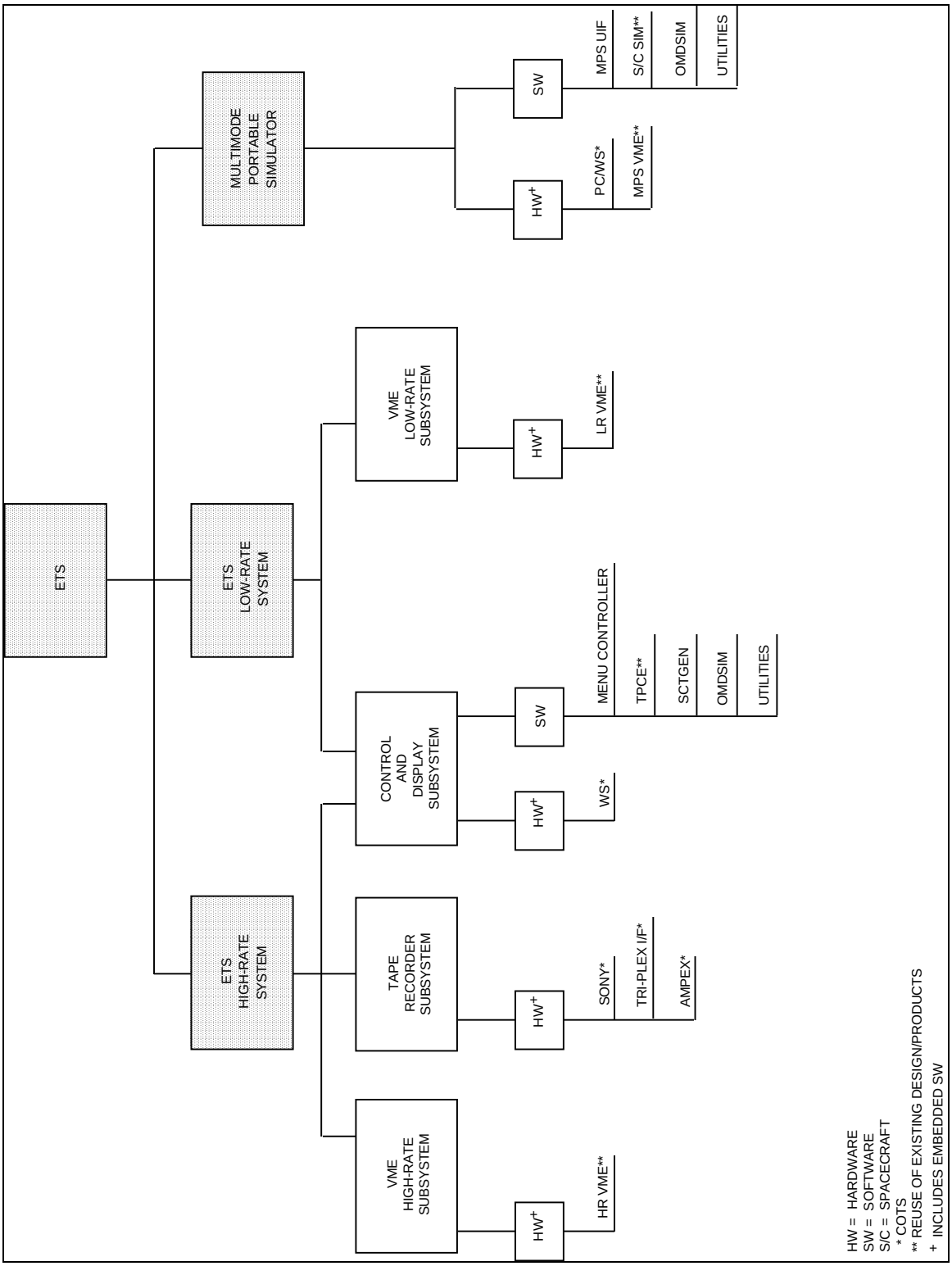

Figure 2-2. System CI Architecture

2.4.2 ETS High-Rate System

The HRS is used for generating and receiving high-rate data, which for ETS includes science data in raw and processed formats. Specifically, the HRS provides the following functions:

- Simulate TGT transmission of up to two 150 Mbps data streams for input to EDOS
- Simulate EDOS transmission of data sets to test interface with DAACs
- Simulate DAAC reception of data sets to test interface with EDOS
- Process SCITF-recorded spacecraft data to generate EDOS compatible data sets

The HRS is composed of three subsystems: the VME High-Rate Subsystem (VHS), the Tape Recording Subsystem (TRS), and the CDS. The VHS transmits data generated by the SCTGEN utility or obtained from external sources, such as instrument or spacecraft test teams. The VHS transmits return-link data in the form of CADUs and EDOS data sets, as well as receives EDOS-generated data sets, depending on the test configuration. The TRS is used to bring in actual spacecraft data captured during thermal vacuum and other testing of the AM-1 spacecraft for playback through the EOSDIS ground system either directly or after modification by using ETS utilities. The TRS can also provide a data archive function. The CDS provides an automated operational environment through which the user controls and monitors the subsystems. It provides capabilities to simulate OMD, generate test data files, manipulate user-provided test data, and assist users in data analysis and report generation functions.

The HRS requirements specifications, excluding those of the CDS support software CIs, are contained in Volume 3. The HRS CDS support software requirements are contained in this volume.

2.4.3 ETS Multimode Portable Simulator

The MPS is a low-fidelity AM-1 spacecraft simulator that provides a capability to test low-rate forward- and return-link interfaces to EDOS and the EOC. The MPS provides the following functions:

- Generate and transmit low-rate spacecraft data as CADUs in serial bitstream (for input to EDOS), as CADUs in Nascom 4800-bit blocks (for input to EBnet as if from a contingency site), and as packets in EDOS data units (EDUs) (for input to EOC)
- Receive and verify spacecraft commands as serial bitstream CLTUs, CLTUs in 4800-bit blocks, or in EOC-EDOS command data blocks
- Use AM-1 project data base (PDB) to generate telemetry and verify commands
- Update telemetry verifiers in real time in response to valid spacecraft commands
- Simulate all AM-1 low-rate data formats, including housekeeping, health and safety, diagnostic and dump
- Simulate two return-link channels simultaneously

- Simulate EDOS rate buffered data file transfers to EOC using MPS-generated data
- Simulate EDOS service header and ground message header fields
- Simulate CODAs using OMDSIM

The MPS has a front-end user interface component and a back-end VME component. The front-end system provides the user interface, and hosts the OMDSIM and other support software CIs. The VME component hosts the spacecraft simulator functions. The MPS detailed design specifications are contained in Volume 2.

2.5 Design Methodology

The OMDSIM and Support Utilities CIs have been developed in accordance with the SEAS System Development Methodology (SSDM). SSDM development approach occurs in four phases: requirements definition, design, implementation and qualification testing, each yielding intermediate products that can be assessed to assure the quality of the development effort. Interactive Development Environment, Incorporated's Software through Pictures (StP) Computer Aided Software Environment (CASE) tool was employed when using structured analysis techniques to decompose functional requirements.

2.5.1 Requirements Definition

The ETS Level 3 Requirements were analyzed, identifying requirements that were allocated to software that provided OMD processing, and utilities that supported other ETS hardware and software functions. Based on the allocated requirements, functionality was defined in further detail, yielding the ETS System Requirements Specification, Volume 1, Reference 4.

2.5.2 Design

Using the ETS SRS, dated October 1995, requirements were allocated to processes and data inputs and outputs were defined in a data dictionary. Processes were defined in data flow diagrams (DFDs), and structure charts developed using the StP CASE tool. An iterative approach was used whereby, lower level processes were defined as needed to assure an unambiguous design. Balancing, or reconciliation of inputs, outputs and process names, of DFDs at all levels was performed using the StP tool.

Figure 2-3 shows the conventions used for structure charts to illustrate the control hierarchy of the modules or units comprising a software item. Each process or spawned subprocess is depicted on it's own page.

Each of the rectangles on a structure chart is a module (a C-language function). Each module has a name field and a brief description

Connections between units are shown as straight lines.

A library unit is represented as a module rectangle with bars on both side.

A named rectangle with a dark triangle attached is an included header file.

A named pentagram represents an off-page connector.

A named open circle with an attached arrow represents incoming or outgoing data. Each name field may contain a list of data elements described by the symbol.

A named closed circle with an attached arrow represents control data.

A shaded diamond symbol represents a selection operation between modules.

A curved arrow, most often used in conjunction with the selection symbol, represents iteration.

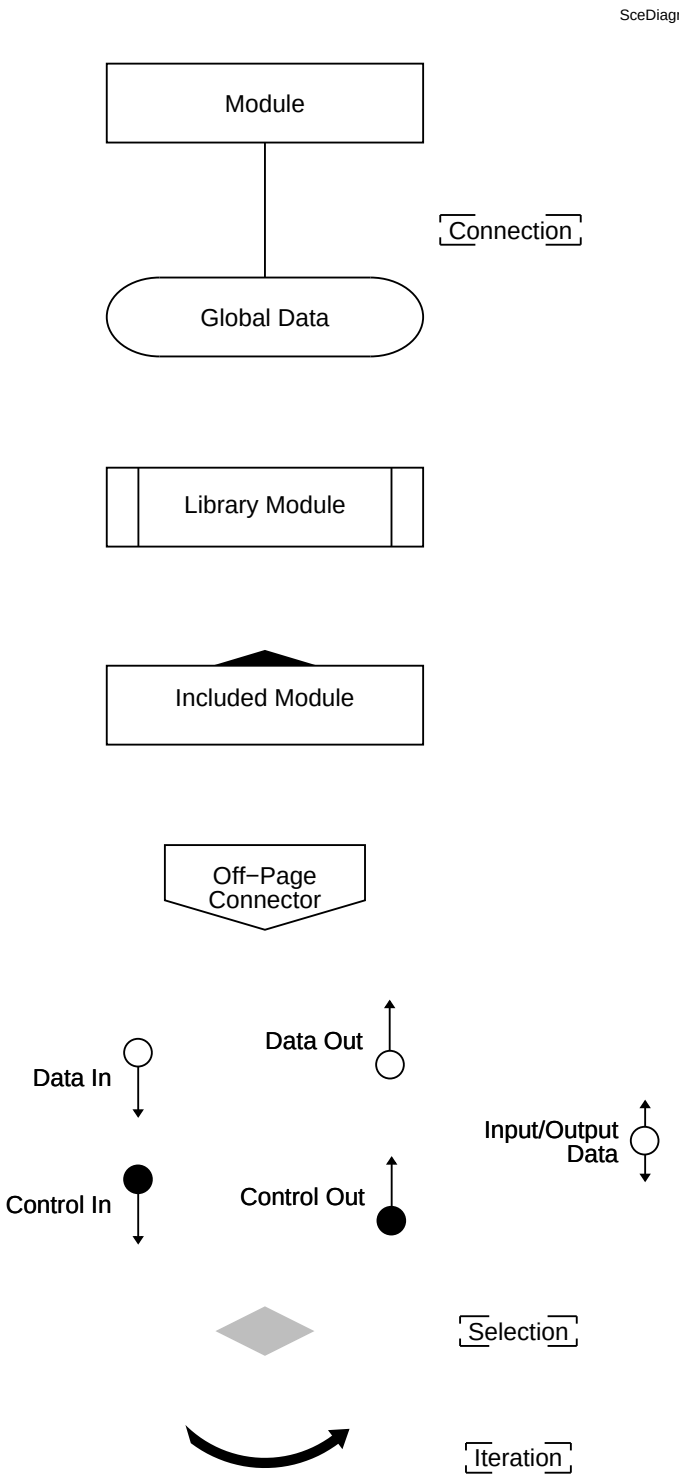

Figure 2-3. Structure Chart Conventions

2.5.3 Implementation

Software will be implemented in the ANSI standard C Programming Language, using an incremental build approach, within the environment described in Section 2.7, Development Environment. Program Design Language (PDL) will be developed as an intermediate product which enables verification of control logic flow. All software functions will contain a prologue which provides the function's name, purpose, invocation, arguments, return values, modification history and any design assumptions or dependencies. All software functions will be tested according to unit test plans. Coded and tested software functions will be inspected through peer reviews which when complete certify that the function is a valid implementation of the design, and satisfies allocated requirements.

The user interface widgets (windows and window contents) will be generated using the Integrated Computer Solutions, Inc. (ICS) BuilderXcessory graphical user interface builder tool. BuilderXcessory generates user interface language (UIL) which can be read and implemented by C language functions; or it can be translated into C language source code. The callback functions for the user interface will also be written in the ANSI standard C Programming Language.

Requirements have been assigned to builds which have been coordinated with other ETS CIs to provide functionality allocated to the build. Functions which implement the allocated design components and requirements will be delivered to an independent team for integration and system testing. A Build Readiness Review (BRR) will be conducted to assure that allocated requirements are met in the build, identify software and hardware units included in the build, and document design modifications that occurred since the ETS Final Design Review.

2.5.4 Qualification Testing

Independent teams will perform integration, system, and acceptance testing of the ETS. The integration test team will assure that CIs comprising ETS interface properly. The system test team will verify that the ETS satisfies requirements identified in the ETS SRS in the test environment. ETS developers will be available to resolve problems that may be identified during testing. The acceptance test team will verify that the ETS satisfies all allocated requirements in the operational environment, and will develop operational procedures and training courses.

2.6 Design Considerations

The major drivers, design modifications since the ETS Design Review, and assumptions used in the detailed design process are documented in this section.

2.6.1 Design Drivers

A major driver for the method of implementation of data exchange between OMDSIM and Support Utilities and other CIs is the extensive code reuse employed by the ETS. Existing TPCE code has established the TPCE parameter servers as the agent for exchanging data.

2.6.2 Design Modifications since the ETS Design Review

The Support Utilities CI provides the capability to log to disk all command blocks received by the CDS workstation from the VLS.

The MPS-hosted OMDSIM processes will be initialized separately from the main MPS process. The transmission and receipt of OMD can be initialized independently from other MPS-hosted functions and CIs.

2.7 Development Environment

The OMDSIM, and Support Utilities CIs will be developed for an HP Apollo Series 700 Model 755 workstation running the HP-UX operating system with 128 MB random access memory (RAM) and 2 gigabyte (GB) internal hard disk. Developers have access to the TPOCC development facility via X-terminals. The target system is located in the CSC ETS development facility and also includes a 4 mm tape drive, a compact disk read-only memory (CD-ROM), a digital audio tape (DAT), a 3.5 inch diskette drive, a 19 monitor, a standard keyboard, and a 3-button mouse.

For build 1, the CIs will be hosted on the LRS CDS workstation and integrated with the other LRS components that reside or were developed in the DSTD lab. For build 2, the LRS CIs, developed in the DSTD and CSC development labs will transported to the SOC for integration testing.

Section 3. Software Configuration Item Detailed Design Descriptions

3.1 Main Menu Controller

3.1.1 Description

The Menu Controller software provides a framework for the various components of ETS: LRS, HRS, and MPS. It displays graphical icons for each ETS system.

3.1.2 User Interface Design

The ETS Menu Controller can be invoked in two ways. If the user is logged in under the Hewlett-Packard Visual User Environment (HP VUE), the ETS icons are available from the personal toolkit located on the graphical Front Panel. There are separate icons for the ETS Low-Rate System (LRS), ETS High-Rate System (HRS) and the ETS Multimode Portable Simulator (MPS). Double clicking on an ETS icon invokes its execution. Alternatively, the user can start ETS in an xterm window by typing “ets_lrs&”, “ets_hrs& or “ets_mps&” as a command line entry.

The Menu Controller software presents the list of ETS software component icons appropriate to the selected system:

- For the LRS, the list includes an LRS TPCE icon, an OMDSIM icon, and an SCTGEN icon.
- For the HRS, the list includes an HRS TPCE icon, an OMDSIM icon, an SCTGEN icon, a Data Comparison Tool icon, and a Data Verification Tool icon.
- For the MPS, the list includes an MPS icon, an OMDSIM icon, and a Project Data Base Conversion icon.

A sample ETS LRS Main Window is shown in Figure 3-1. Each icon represents a separately executable software component. To activate an application, click once on its icon to bring up the icon menu list and then click on the *Restore* option; or double click on the icon. Most of these icons, such as the TPCE-based ones, can only be invoked for one execution at a time due to fixed hardware resources. After a limited execution icon has been activated, the *Restore* option will be dimmed on its icon menu.

To terminate execution of a software component, the preferred method is to use the *Exit* option from the main window of that component. An alternative method is to click once on the icon again to bring up the icon menu list. The *Close* option may be selected to send a terminate signal to the component. The termination sequence for that component will then be started. This may involve querying the operator in order to save data files and confirming the termination request.

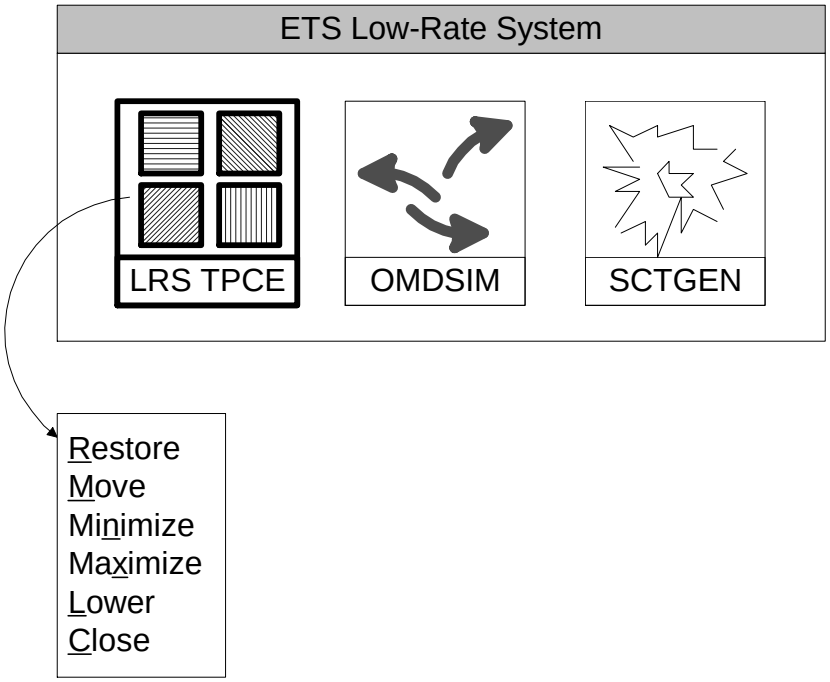

Figure 3-1. Main Window for ETS LRS Menu Controller

3.1.3 Processing

The processing of the Menu Controller is essentially that of an icon box as defined by Motif 1.2. The Menu Controller displays the set of icons appropriate to the given system. When icons are selected, the user may activate or terminate the represented application. For applications that are dependent on fixed hardware resources, multiple executions are not permitted. After applications have been started, selecting their icons provides a method for requesting application termination.

3.1.4 Prologs/Program Design Language/Structure Charts for Module Drivers

Prolog/PDL descriptions are not generally provided for graphical user interface design. These designs generally include diagrams of the interface elements (widgets) and text describing the actions performed by user events in the widgets. Callback functions which are called in response to events perform some widget maintenance as well as provide the connection between the user interface and the application. Their design is specified in the widget/action description.

A structure chart for Menu Controller software is shown in Figure 3-2 and a brief overview of the routines is provided in Table 3-1.

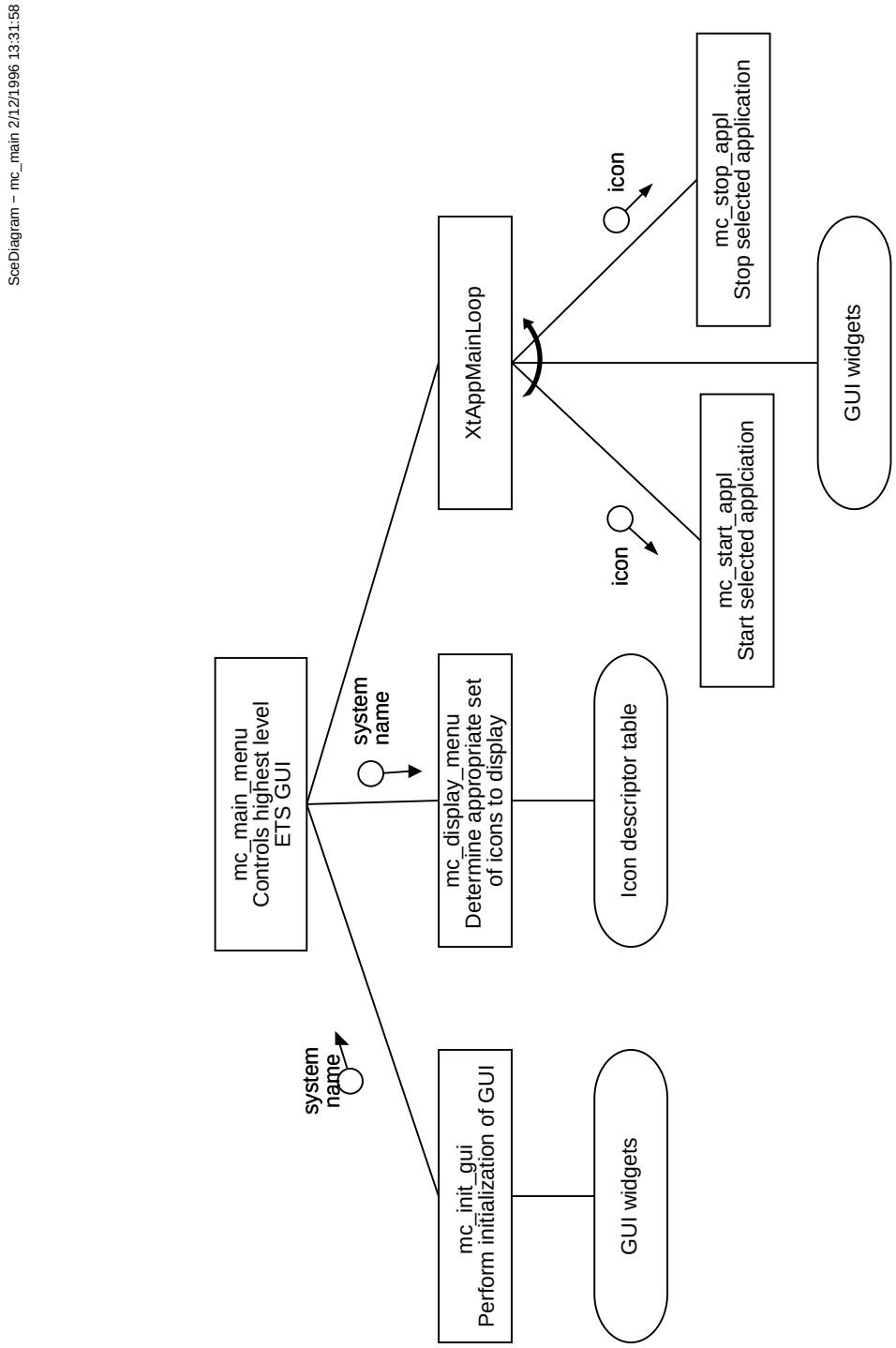

Figure 3-2. Structure Chart for Menu Controller

Table 3-1. Menu Controller Software Routines

Name	Description
XtAppMainLoop	This is the X Toolkit Intrinsics routine that transfers control of the X window application to Xt and interacts with application callback routines as needed.
mc_display_menu	This routine accesses the icon descriptor table to display the appropriate set of icons for the given system.
mc_init_gui	This routine performs initialization of the menu controller task. This involves the usual setup for the gui environment as well as determining the ETS system to be controlled.
mc_main_menu	This is the main routine for the menu controller task.
mc_start_appl	This routine attempts to start up the requested application. In addition to checking for limited execution status, it is also possible that a given icon has already been activated for another ETS system.
mc_stop_appl	This routine attempts to terminate the selected application by sending the termination signal.

3.2 OMD Simulator

3.2.1 Description

The OMD Simulator is a stand-alone process that provides a source of OMD messages to supplement ETS command and telemetry data streams. In particular, the OMDSIM CI supports Customer Operations Data Accounting (CODA) reports, TDRSS Summary Session (TSS) reports, EDS/PDS Delivery Records, and EDS/PDS Acceptance Notifications.

Although OMDSIM is capable of operating without the support of other ETS components, it is designed to accept signals via TPCE that indicate the availability of data for transmission (CODA information), the end of the simulated TDRS session and the associated request for TSS report transmission. When desired, the operator can use the user interface to establish the values of the data fields comprising the OMD, indicate a transmission schedule (interval between transmissions and duration of the schedule) for these prepared OMD, and generate the signals to begin and terminate the testing phase of operations (that is, signals that emulate the actions of TPCE as indicated above).

In addition, OMDSIM:

- accepts OMD from other EOSDIS test elements and logs their receipt
- logs the event of transmission of OMD
- logs the OMD transmitted and received to a file

- displays the events of transmission and receipt of OMD
- displays OMDSIM status and anomalous occurrences
- maintains, displays and logs OMD accounting statistics
- responds to received OMD (responses TBD)

Certain relevant OMD message items, such as the number of CADUs received, are expected to change over time. To simulate this when other ETS elements are not being used to provide values that actually reflect the associated data sets, OMDSIM uses ramping functions, parameterized by constants at initialization. The operator is permitted to designate that these values are to be calculated for the next OMD to be transmitted or allowed to remain static. For certain transmitted values there are three possible sources: the user provided static (or default) value, the "real-time" value provided by other ETS components that are meant to accurately reflect the associated data sets (if available in the present configuration) and calculated values meant to simulate to low fidelity the associated data sets.

Throughout the OMDSIM user interface and processing sections of this document, the definitions outlined below have been adopted.

> The term "loading a scenario" refers to the act of sending the scenario filename to the om_scenario_mgr subprocess by the user interface.
>
> The term "loading an OMD" refers to the act of writing an OMD to the omd queue by the user interface.
>
> "Transmission Test" refers to the activity that is started when the **Start Transmission** button is clicked on. An OMD is loaded and transmitted repeatedly according to the interval and duration specified in the user interface. The OMD is modified with each successive transmission according to the directives specified in the working_OMD structure. The test ends when the operator clicks on the **Stop Transmission** button or the test duration has expired.
>
> The scenario file is comprised of records, each containing a transmission interval, transmission duration for the OMD type, the OMD type indication and a description of the OMD (in working_OMD) to be transmitted. The transmission interval indicates the number of seconds after the previous transmission (or the start of the scenario execution) to send the currently loaded OMD. The test duration may be 0 to indicate only one transmission of the loaded OMD, greater than 0, or "infinite", to imply that the loaded OMD will be transmitted every interval seconds until a stop signal is received.
>
> "Scenario Execution" refers to the activity that is started when the **Execute Scenario** button is clicked on. A scenario file is loaded and executed according to its directives. Scenario execution ends when the operator clicks on the **Terminate Scenario** button or an end of file is encountered.

3.2.2 User Interface Design

OMDSIM can be invoked either through the Menu Controller or by the UNIX command

% omdsim XXX

where XXX is “LRS,” “HRS,” or “MPS” for the system OMDSIM is to be connected to. The Menu Controller actually uses this command to invoke OMDSIM.

3.2.2.1 Definitions and Conventions

The conventions used in this section are defined in section 1.5 of this document.

3.2.2.2 OMD Simulator Main Window

The Main Window, shown in Figure 3-3, appears when the OMD Simulator is invoked. It contains the primary controls of the system.

When operating with LRS, the OMD Simulator is limited to generating only CODA Report and TSS Summary OMD, parts of which are obtained from LRS. Parameters not given by LRS are taken from the default CODA Report and TSS Summary which are established by the operator. OMD transmission is driven by operator directives from the user interface via the **Start/Stop Transmission** button and the transmission interval, test duration, and TSS Summary delay fields.

Scenario files cannot be executed when the external system is LRS because the LRS is to be a “realtime” system with no simulation required. The generated OMD reflect the actual processed forward- and return-link data. Therefore the Scenario File label and field and the **Execute Scenario** button are all desensitized. The File options that involve scenario operations, *Load Scenario...*, *New Scenario...*, and *Open Scenario...*, are desensitized.

While operating with MPS and HRS, scenario files can be loaded and executed and any OMD type can be loaded and transmitted. The parameters for the OMD can be modified by the operator.

Other than these differences, OMDSIM looks the same for whatever external process it operates with.

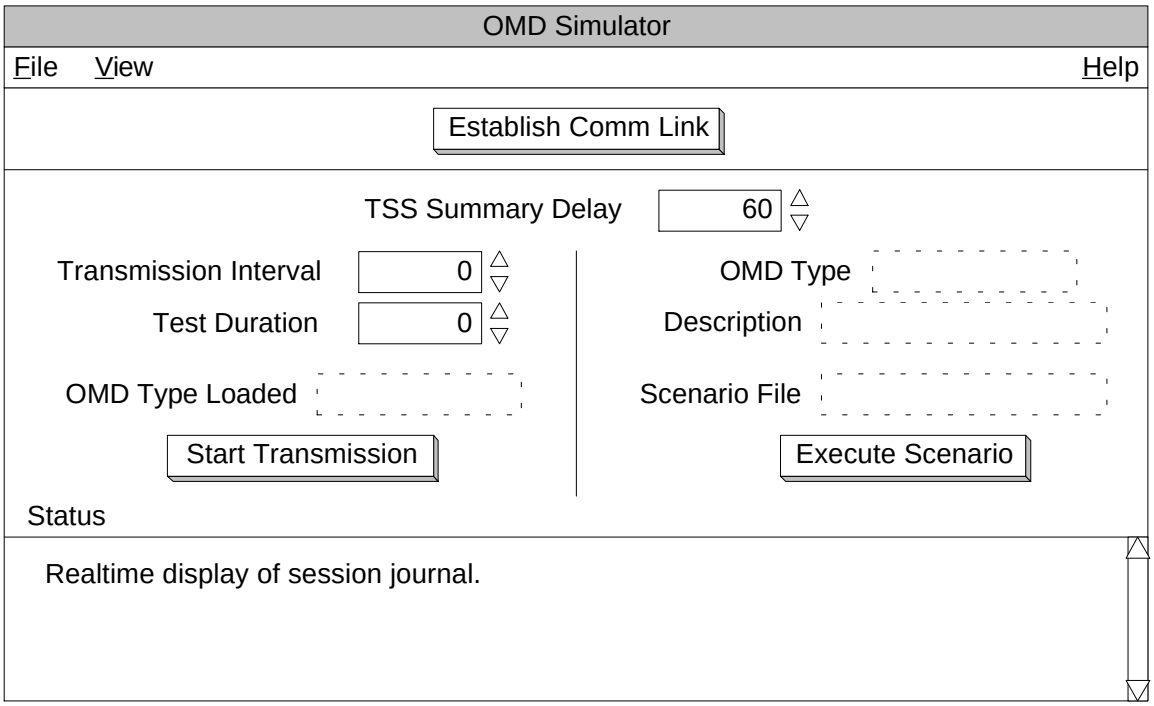

Figure 3-3. OMD Simulator Main Window

Selecting the **Establish Comm Link** button

checks all communications ports;

enables communications with external systems (EBnet and LRS, HRS, or MPS); (If connection isn't established within a TBD timeout period, a question dialog box pops up with the message "Unable to establish communications link with XXX. Continue without XXX?". Where XXX is the name of the other system. Clicking on OK allows OMDSIM to continue independently of the other system. If the communications link with EBnet cannot be established, OMDSIM cannot transmit or receive OMD; hence it continues operating in the pretest mode and the callback for Establish Comm Link exits here.

starts up the all OMD subprocesses;

opens fifos shared with subprocesses;

and sensitizes the **Start Transmission** button and, if communicating with HRS or MPS, the **Execute Scenario** button.

The **Establish Comm Link** button is a toggle; its label changes to **Break Comm Link** while the link is active. Selecting **Break Comm Link**

asks the operator via a confirmation dialog box if the activity in progress should be stopped;

breaks communications with external systems;

closes fifos shared with subprocesses;

stops subprocesses;

and, desensitizes the **Execute Scenario** button, the **Start Transmission** button, the Load OMD option, and the Load Scenario option. (OMD cannot be transmitted when the communications link is broken.)

For Transmission Tests, the transmission interval, test duration, and TSS Summary delay are set and used in the same way regardless of OMDSIM's external partner.

The OMD transmission interval is set by selecting the spinners (arrows) to the right of its field or by editing the field directly. If the transmission interval is 0, the Test Duration field is desensitized and only one OMD is sent.

The Test Duration is set by clicking on the arrows to the right of its field or by editing the field directly. If the test duration is set to 0, only one OMD is transmitted. The spinners are programmed so that selecting them past some time limit (TBD) causes the test duration to be infinity. This time limit can be overridden by editing the field directly.

The TSS Summary Delay is set by clicking on the arrows for the TSS Summary Delay or by editing the field directly. This specifies the length of time after the last OMD transmission that the TSS Summary is transmitted. The default is 60 seconds.

3.2.2.3 Scenario Execution

If a scenario file has been loaded, its name is displayed in the Scenario File field. Both the label and field are dimmed until the scenario is executed. After execution, the scenario is unloaded and a scenario must be loaded again before another scenario can be executed.

Clicking on **Execute Scenario**

changes the button's label to **Terminate Scenario**;

sends the TSS Summary delay to om_scheduler:

sends a start message to om_scheduler;

sensitizes the Scenario File label and field, the OMD Type label and field and Description label and field;

and disables the **Start Transmission** button, the Transmission Interval field, the Test Duration field, the TSS Summary Delay field and the *Load Scenario...* File options in the Main Window and the Scenario Editor.

The **Terminate Scenario** button

sends a stop message to om_scheduler;

desensitizes the Scenario File label and field, the OMD Type label and field, and the Description label and field;

sets the allow_transmission_test flag to true;

sensitizes the TSS Summary Delay field, and the *Load Scenario...* File options in the Main Window and the Scenario Editor;

and changes the button's label back to **Execute Scenario and desensitizes it**.

Briefly, the om_scenario_mgr subprocess reads each record one at a time from the scenario file. It sends the schedule information (interval and duration) to om_scheduler and puts the OMD into the OMD queue and waits until it receives the ready_for_next_record message from om_scheduler. This is repeated for every record until an EOF is encountered or om_scenario_mgr receives a stop message (from om_scheduler). Om_transmission_task reads the OMD from the OMD queue, modifies it if necessary, and transmits it according to the signals it receives from om_scheduler. Om_transmission_task transmits a TSS Summary and task_complete message to the main UI task in the same way it does for a transmission test. Om_scenario_mgr ends scenario execution either by executing through the last scenario record or by receiving a stop message from om_scheduler.

3.2.2.4 Transmission Test

If an OMD is loaded into memory ready for transmission, its type is displayed in the OMD Type Loaded field. Both the label and field are gray until an OMD transmission test is started. The **Start Transmission** button is desensitized until an OMD is loaded. After the transmission is done, an OMD must be loaded again to perform another transmission test. After loading an OMD, the operator can start transmission, via the **Start Transmission** button or unload the OMD, via the *Unload OMD* option in the File menu.

Clicking on **Start Transmission**

changes the button's label to **Stop Transmission;**

sends the transmission interval, the test duration, and the TSS Summary delay to om_scheduler;

sends a start message to om_scheduler;

sensitizes the OMD Type label and field;

and disables the Transmission Interval field, the Test Duration field, the TSS Summary Delay field, and the *Load OMD...* File options in the Main Window and OMD Editor.

Clicking on **Stop Transmission**

sends a stop message to om_scheduler;

clears the OMD Type loaded field and desensitizes the OMD Type Loaded label and field;

enables the TSS Summary Delay field, and the *Load OMD...* and, if operating with LRS, the *Load Scenario...* File options in the Main Window and OMD Editor and Scenario Editor;

and. changes the button's label to **Start Transmission** and desensitizes it**.**

3.2.2.5 Main Window Menu Bar

Following are the menus and their options found on the menu bar of the Main Window.

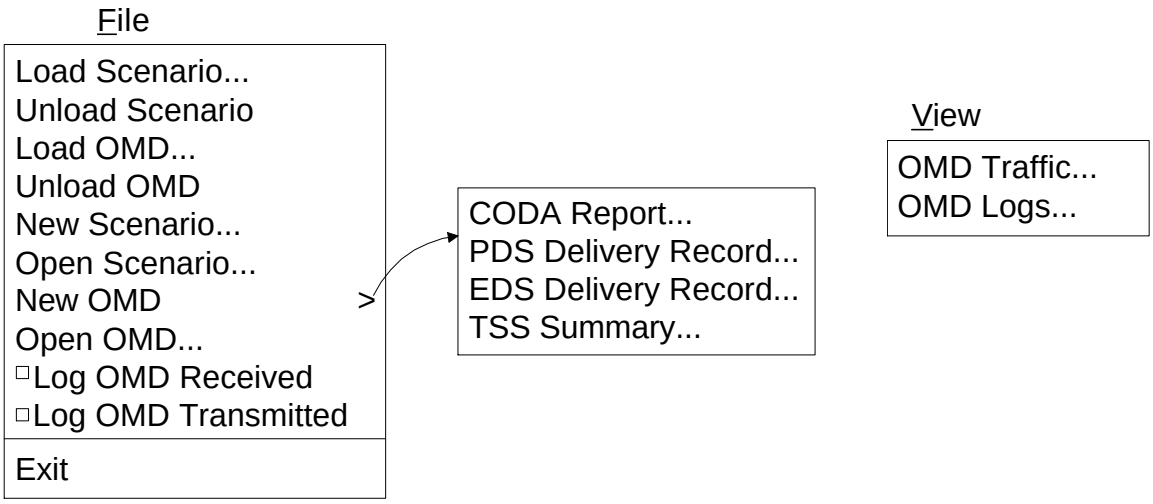

Figure 3-4. Main Window Menu Bar

3.2.2.5.1 File

The File menu contains the following options:

Load Scenario... causes a file selection box to appear. Selecting a scenario file from the list displayed and clicking on **OK** loads the scenario. This option is desensitized while a transmission test is active or a scenario is being executed or if the external system is LRS.

Loading a scenario (by selecting on *Load Scenario...* under the File menu or *Load* in the Scenario Editor under File)

displays the scenario filename in the Scenario File field;

sends the scenario filename to the scenario subprocess;

sensitizes the *Unload Scenario* File menu option;

desensitizes all OMD operation options;

and sensitizes the **Execute Scenario** button.

Unload Scenario unloads the current scenario by

sending an Unload_scenario message to the om_scheduler subprocess;

sensitizes the *Load Scenario...* option and all options involving OMD operations;

and desensitizes the Scenario File label and field.

Load OMD... causes a file selection box to appear. Selecting an OMD file from the list displayed and clicking on **OK** loads the selected OMD. This option is desensitized while a transmission test is active or a scenario is being executed.

Loading an OMD (by selecting *Load OMD...* under the File menu or *Load* in the OMD Editor File menu)

displays the type of OMD loaded in the OMD Type Loaded field;

puts the working_OMD into the OMD queue;

sensitizes the Unload OMD File option;

desensitizes all options involving scenario operations;

and sensitizes the **Start Transmission** button.

Unload OMD unloads the current OMD by

sending an Unload_OMD message to the om_scheduler subprocess.

sensitizes the *Load OMD...* option and all options involving scenario operations;

and desensitizes the OMD Type Loaded label and field.

New Scenario... brings up the Scenario Editor which allows the operator to create a new scenario file. This option is desensitized if the external system is LRS.

Open Scenario... pops up a file selection box. Selecting a file and clicking on the **OK** button brings up the Scenario Editor initialized with the values in the selected scenario file. This option is desensitized if the external system is LRS.

New OMD > brings up the OMD Editor for the OMD type selected in the cascade menu to allow the operator to create a new OMD file. The TSS Summary type of OMD can be edited (is sensitized) only for MPS; otherwise, TSS Summary information is accumulated during operations with LRS and HRS.

Open OMD... causes a file selection box to appear. Selecting a file and clicking on the **OK** button brings up the OMD Editor initialized with the values in the selected OMD file.

Setting the *Log OMD Received...* option brings up a file selection box to allow the operator to enter a filename and directory for the log. The extension, ".omd_rcvd", is automatically appended. The default filename is timestamp.omd_rcvd, where the timestamp reflects the time the file is created. Selecting a file and clicking on **OK** opens the file and sets a flag for OMDSIM to log OMD received. Unsetting this option resets the flag so OMDSIM stops logging OMD received and closes the file.

Setting the *Log OMD Transmitted...* option brings up a file selection box to allow the operator to enter a filename and directory for the log. The extension, ".omd_sent", is automatically appended. The default filename is timestamp.omd_sent, where the timestamp reflects the time the file is created. Selecting a file and clicking on **OK** opens the file and sets a flag for OMDSIM to log OMD transmitted. Unsetting this option resets the flag so OMDSIM stops logging OMD transmitted and closes the file.

Selecting *Exit* shuts down the OMD Simulator. If there are any tasks active, a confirmation box pops up with the message, "There are tasks still active. Shutting down OMDSIM will stop all tasks. Continue with shutdown?". Clicking on **Yes** in the confirmation box stops all activity in progress and continues with the exit procedure. The communications links are broken and all files are closed. Clicking on **No** returns control to the OMD Simulator's main window. **No** is the default.

3.2.2.5.2 View

OMD Traffic... pops up a dialog box which displays the date and time and OMD type transmitted and received for realtime monitoring of OMD Traffic.

OMD Logs... pops up a file selection box. Selecting a file and clicking on **OK** pops up a dialog box similar to the OMD Editor. Received OMD logs and transmitted OMD logs can be viewed through this option. The OMD log files are binary. The diagram for this dialog box is TBD.

3.2.2.6 Scenario Editor

The Scenario Editor window appears when the operator clicks on *New Scenario...* or selects a scenario file to edit from the *Open Scenario...* option in the File menu.

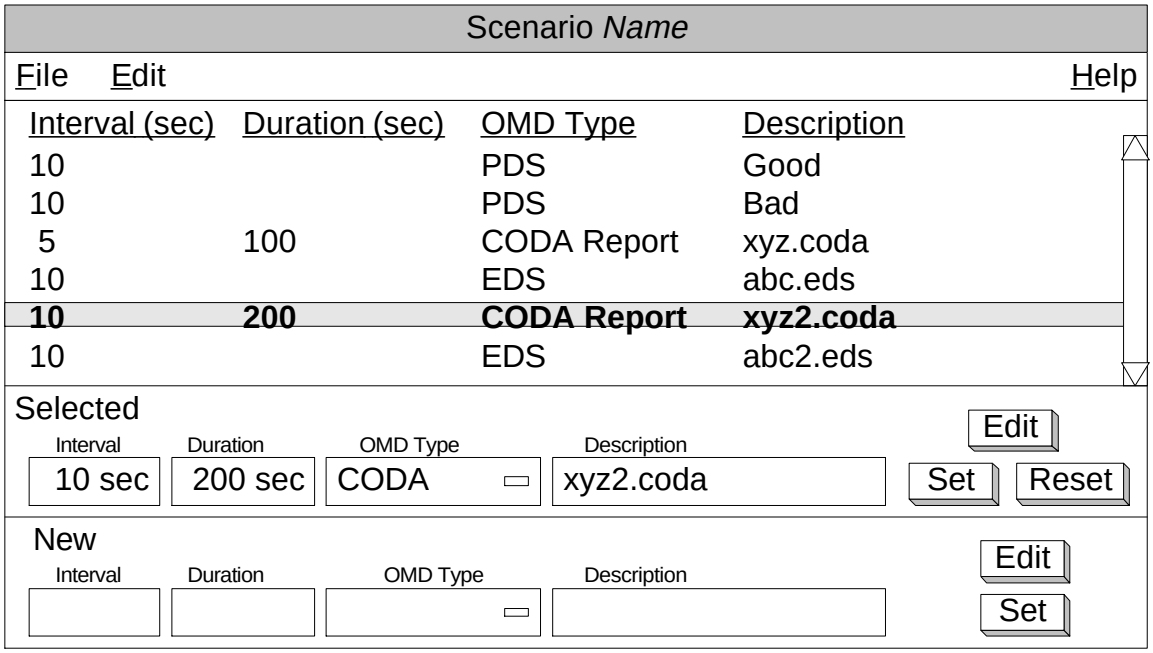

Figure 3-5. Scenario Editor

The scenario file is composed of records. Each record gives the transmission interval, test duration, the OMD type, a description (of the OMD), and the actual OMD to be transmitted. The transmission interval gives the time interval (format TBD) after the previous transmission (or after the start of scenario execution) to send the current OMD. The test duration is optional. If it is not specified or is zero, only the current OMD is transmitted. If the test duration is specified and is greater than zero, OMD are transmitted repeatedly at the interval specified until the duration expires. Negative numbers for these values are not valid and are treated as if they were zero. Depending on how the OMD in the record is set up (see OMD Editor), its counters will be static or automatically calculated for each succeeding transmission. The calculations are done according to algorithms specified for each counter for each OMD type in the omd_sim_params file that is read during initialization of the om_transmission_task subprocess. The description in the record contains a brief 80-character comment.

In the Scenario Editor, the transmission interval, test duration, the OMD type, and description for each record in the scenario is displayed in the scrollable list. OMD can only be viewed or edited through the OMD Editor which can be brought up while in the Scenario Editor. Selecting a record puts the interval, duration, OMD type, and description in the "Selected Record" section near the bottom of the dialog box. The OMD type is given as the current OMD type in the OMD Type option menu.

The "Selected Record" section allows the operator to modify an existing record. The Transmission Interval, Test Duration, and Description are changed by editing those fields. The OMD type can be changed by selecting from the OMD Type Option menu. If the type is changed, the OMD editor is brought up and initialized with the default OMD of that type. The operator can then modify this OMD. If the OMD type is not changed, clicking on the **Edit** button to the right of the OMD type field or double selecting the scenario record in the list brings up the OMD Editor containing the OMD in the selected record. If the OMD is modified (or new), it must be saved for the scenario by selecting *Update Scenario* in the OMD Editor's File menu. OMD edited for a scenario file can also be saved in OMD files. Clicking on **Set** changes the selected record as specified in this section. Clicking on **Reset** restores the record to its previously stored state.

The "New Record" section accepts information for a new record, which is inserted immediately after the selected record, or appended to the end of the file if no selection is made. The Transmission Interval, Test Duration, and Description are specified by editing those fields. The OMD type can be specified by selecting from the OMD Type Option menu. The default OMD type is the one previously specified. Changing the OMD type or clicking on **Edit** pops up the OMD Editor initialized with the default OMD of the type specified. Select *Update Scenario* in the OMD Editor's File menu to put the modified OMD into the scenario. Clicking on the **Set** button inserts the new record.

3.2.2.7 Scenario Editor Menu Bar

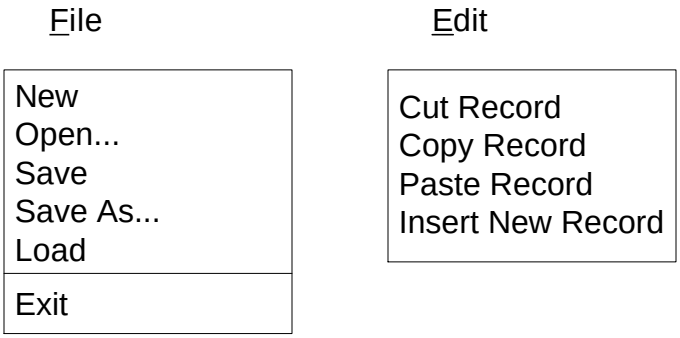

Figure 3-6. Scenario Editor Menu Bar

3.2.2.7.1 File

New clears the Scenario Editor to allow the operator to create a new scenario. New records are taken from the "New Record" section.

Open... brings up a file selection box. Selecting a file and clicking on **OK** reinitializes the scenario editor with the contents of the selected scenario file.

Save saves the current scenario to the same file selected in the *Open...* option or to the file last saved. If the scenario is new, a file selection box pops up to request a filename and location for the new file (i.e. "*Save As...*" is activated).

Save As... pops up a file selection box to request a filename and location in which to save the current scenario. Scenario files have the extension ".scn", which, if not specified by the operator is appended by the software. If the operator specifies an extension, the ".scn" is added after that.

Load writes the current scenario to disk (if the filename is not known, a file selection box prompts the operator for the filename and directory) and loads it for the next scenario execution. The scenario load operation is described in the main window section. This option is desensitized while a scenario is being executed.

Exit closes the Scenario Editor. If the current scenario has been modified but has not been saved, a dialog box pops up to ask the operator to save the scenario. Clicking on **Yes** in the question dialog box saves the scenario as if the *Save* option had been clicked on and continues closing the Scenario Editor. Clicking on **No** in the question dialog box continues to close the Scenario Editor without saving the scenario. Clicking on **Cancel** returns control to the scenario Editor as it was before *Exit* was attempted.

If the current scenario was not loaded and a scenario is not being executed, a dialog box pops up to ask the operator to load the scenario. Clicking on **Yes** loads the scenario and continues closing the Scenario Editor. Clicking on **No** continues to close the Scenario Editor without loading a scenario. Clicking on **Cancel** returns control to the Scenario Editor.

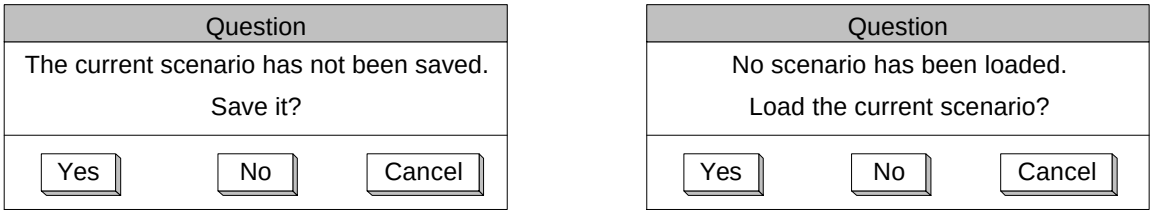

Figure 3-7. Scenario Editor Confirmation Boxes

3.2.2.7.2 Edit

While editing a scenario, only one record can be selected at a time.

Cut Record copies the selected record into a temporary buffer and deletes it from the scenario. The deleted record can then be pasted to another position in the file. The cut record option is dimmed until a record is selected. After the cut, the selection indicator is placed on the record before the record that was removed.

Copy Record copies the currently selected record into a temporary buffer. Only one record can be copied at a time. The *Copy Record* option is dimmed until the operator selects a record.

Paste Record copies the record from the temporary buffer into the position after the record selected. If no record is selected, the OMD is appended to the end of the scenario. The *Paste Record* option is dimmed until a record has been copied or cut. After the paste, the newly pasted record becomes the selected record.

Insert New Record allows the operator to insert a new scenario record after the one selected. If no record exists, the OMD becomes the first. If no record is selected, the OMD is appended to the end of the scenario. The new record is taken from the "New Record" section of the dialog box.

3.2.2.8 OMD Editor

The OMD Editor window appears when the operator selects an *Open OMD...* or *New OMD >* from the main window's File menu or from the Scenario Editor as described above. If the OMD is new, the Ground Message Header dialog box is popped up first. The header for each OMD is stored with the OMD.

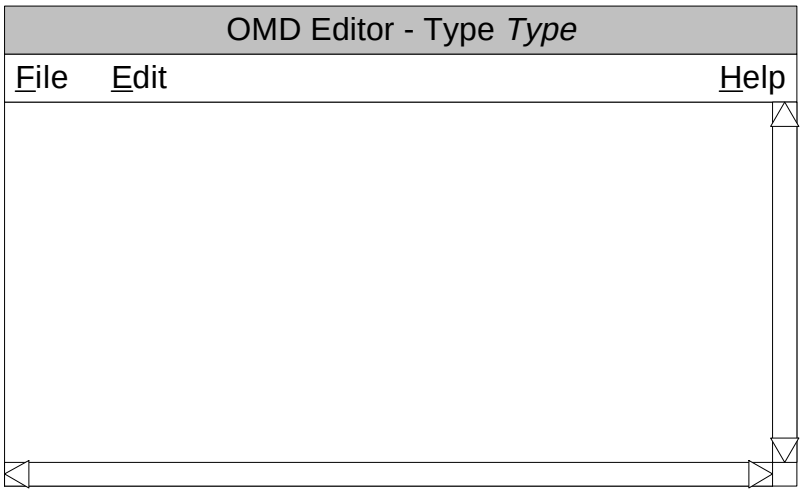

Figure 3-8. OMD Editor

The OMD Editor's display format depends on the type of OMD being edited. The type is determined by the filename's extension, from the OMD type given in the scenario file, or from the type selected under *New >* in the OMD Editor's File menu and *New OMD >* in the Main Window's File menu.

The OMD Editor window consists of panels that display logical or functional parts of the OMD. All the panels combined may be larger than the window that contains them so the window has scroll bars which are used to scroll the full display There is a different collection of panels for each OMD type. The Editor knows which set of panels to use because of the OMD type selected or the extension of the OMD file.

All parameters are validated when the OMD is saved. If an inconsistency occurs, the invalid values are flagged by reverse video and a warning dialog box pops up. The warning is "Invalid values were found in the current OMD. Include all values as specified in the OMD?" Clicking on **Yes** continues with the save operation and closes the warning dialog box. Clicking on **No** closes

the warning dialog box, nothing is saved and control returns to the OMD Editor, where the user can edit the values. The invalid values remain highlighted until the user has changed them.

OMD edited for scenario files can be treated as individual OMD also. That is any operation that can be performed on an OMD from an OMD file or a new OMD can be performed on an OMD from a scenario file.

The OMD Editor remembers the most recently edited OMD even when it has been closed. If invoked with the same type as the last OMD edited, the OMD Editor uses the previous OMD as a template for changes.

3.2.2.9 OMD Editor Menu Bar

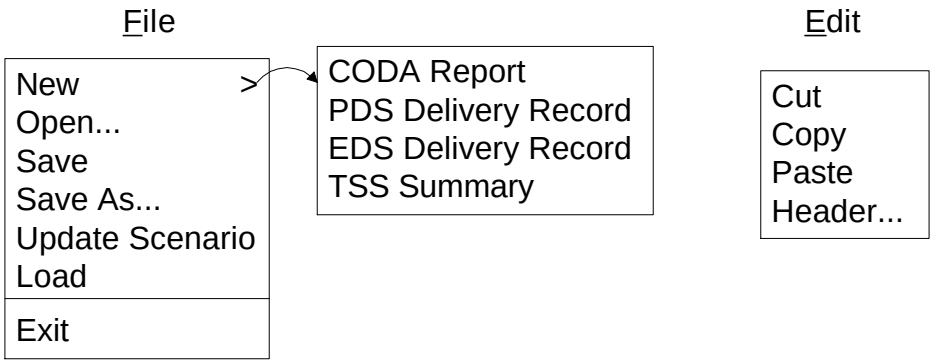

Figure 3-9. OMD Editor Menu Bar

3.2.2.9.1 File

Selecting an OMD type from the cascade menu of the *New* > option initializes the editor with the default OMD parameters of the specified type. If the OMD Editor is invoked from the Scenario Editor, only the OMD type specified there is available in the *New* > cascade menu. . The TSS Summary type of OMD can be edited (is sensitized) only for MPS; otherwise, TSS Summary information is accumulated during operations with LRS and HRS.

Open... brings up a file selection box. Selecting an OMD file and clicking on the **OK** button in the file selection box brings up the OMD initialized with the values from the selected OMD file.

Save saves the current OMD to the same file selected in the *Open...* option or to the file last saved. If the OMD is new, a file selection box pops up to request a filename and location for the new file.

Save As... pops up a file selection box to request a filename and location in which to save the current OMD. OMD filename extensions are the OMD types, ".coda," ".pds," ".eds," and ".tss". The filename root can be anything; the extension is Automatically given by the OMD Simulator.

Update Scenario puts the edited OMD into the scenario record being modified. This option is desensitized if the OMD Editor is invoked from the Main Window.

Load causes the current OMD to be loaded into memory for the next transmission signal. The OMD load operation is described in the main window section. This option is desensitized while a transmission test is active or a scenario is being executed.

Exit closes the OMD Editor. If the current OMD has been modified but has not been saved, the first question box in Figure 3-10 pops up to ask the operator to save the OMD. Clicking on **Yes** in the question dialog box saves the OMD as if the *Save* option had been clicked on; the Scenario Editor is then closed. Clicking on **No** in the question dialog box continues to close the OMD Editor without saving the OMD. Clicking on **Cancel** returns control to the OMD Editor as it was before *Exit* was attempted. Yes is the default.

If the OMD Editor was invoked from the Main Window of the OMD simulator, it is assumed to be independent of a scenario and if no OMD was loaded during the current edit session, *Exit* brings up the second question box in Figure 3-10 to ask the operator whether to load the current OMD. Clicking on **Yes** in the question box loads the current OMD and continues closing the editor. Clicking on **No** in the question box, continues closing the editor without loading an OMD. Clicking on **Cancel** returns control to the OMD Editor as it was before *Exit* was attempted.

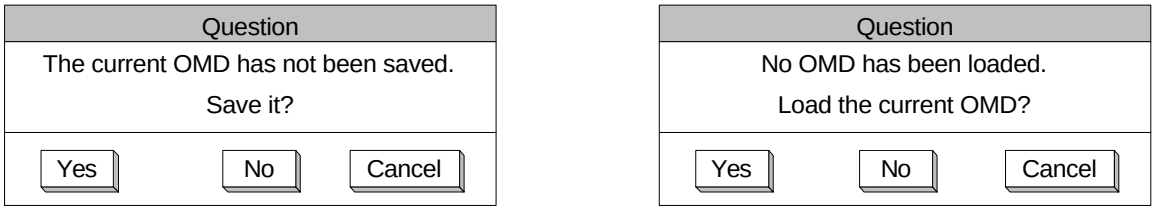

Figure 3-10. OMD Editor Question Boxes

3.2.2.9.2 Edit

Header... pops up the Ground Message Header dialog box initialized with the information given in the current OMD. If the OMD is new, the header is initialized with default information. The default header comes from the default file for the specified OMD type. If the message header is modified or is new, it must be saved for the current OMD by selecting *Save* in the File menu of the Ground Message Header dialog box.

3.2.2.10 CODA Report Editor Panels

In the following panels, all count parameters can be set to be incremented automatically or can be static for each succeeding transmission of the same CODA Report. Selecting **Auto** causes the count to be calculated from a model or ramp. The algorithm for each programmable count parameter is given in the initialization parameters file for OMDSIM. Selecting **Static** causes the count to be the value given in the Static field. The static field is set to a default value or the value given in the OMD that initialized the Editor. The **Auto** option is the default. If OMDSIM is connected with LRS, the om_transmission_task merges the information it receives from LRS with the working_OMD structure it receives from the scenario or user interface. Elements of the OMD that are marked “static” remain static and always have the value given in the static field.

Elements marked “Auto” are replaced by information from the LRS if it is available; otherwise, they are calculated as described above.

3.2.2.10.1 CODA Report Content Indicators

The leftmost top panel contains the CODA Report Content Indicators.

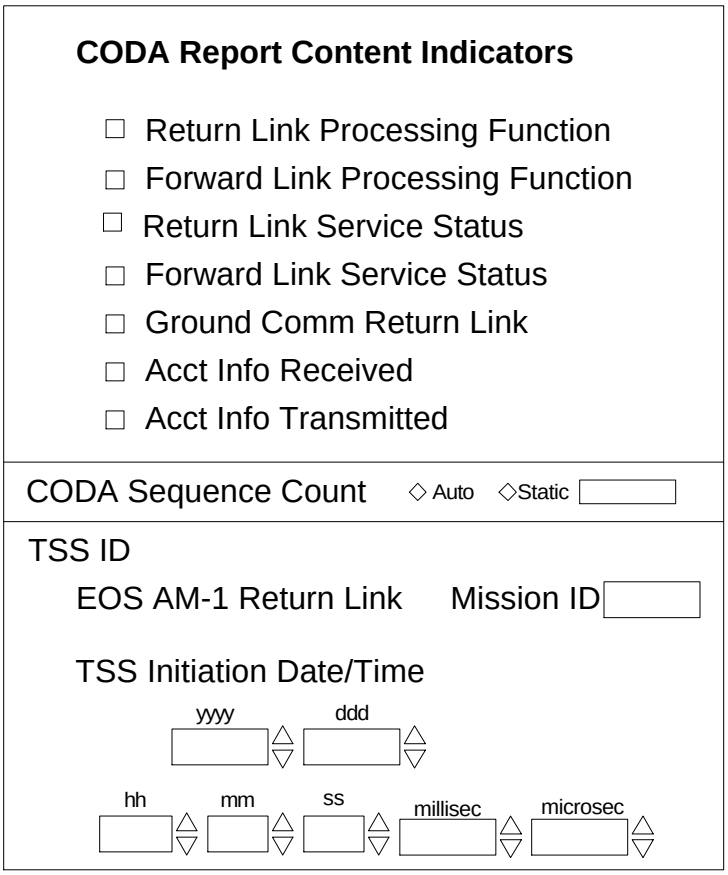

Figure 3-11. CODA Report Content Indicators

The first section determines which of the other panels is available for input. Each option corresponds to a panel. Setting an option causes its panel to be sensitized. Unsetting it causes its panel to be desensitized and any information in it to be ignored; the information in the desensitized panel is kept during the current edit session.

The CODA Report Sequence Count can be set to be incremented automatically or can be static for each succeeding transmission of the same CODA Report. Selecting **Auto** causes the count to be calculated from a model or ramp or given by an external source. Selecting **Static** causes the count to be the value given in the Static field. The static field is set to a default value or the value given in the OMD that initialized the Editor. The **Auto** option is the default.

The TSS ID is a combination of the Spacecraft ID and the Date/Time of initiation. The date is a Julian day and a 4 digit year which can be changed by editing or clicking on the spinner arrows. The hours minutes seconds milliseconds and microseconds can be changed by editing their fields

directly or by clicking on their spinner arrows. Before transmission, the hours minutes and seconds are converted to the second of the day by the om_transmission_task subprocess.

3.2.2.10.2 Return Link Processing Function

The Return Link Processing Function panel is sensitized when the option of the same name is set.

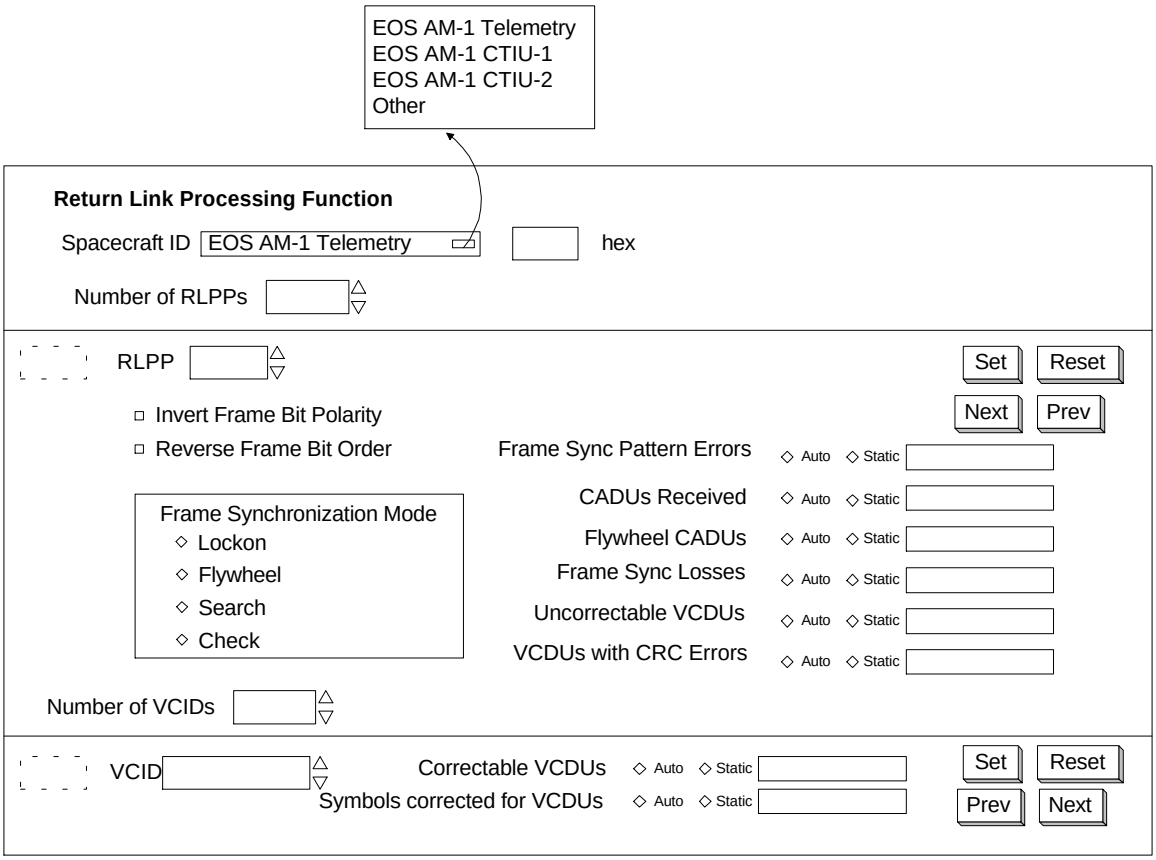

Figure 3-12. Return Link Processing Function

The Spacecraft ID field is editable and a hexadecimal value less than 0xff is required.

The Number of RLPPs determines how many RLPF Target Ports (RLPPs) are in the OMD. The range is 0 to 24.

The ordinal number of the current RLPP is displayed for information purposes in the field to the left of its identification. The RLPP ID is changed by editing its field or clicking on the spinners. The ordinal number cannot be edited.

The number of VCIDs determines how many VCID sections there are. The scroll bar is used to display all VCIDs.

The ordinal number of the current VCID is displayed to the left of the VCID identification. The identification field can be modified by editing or clicking on its spinner arrows. The ordinal number is for information only and cannot be changed.

All VCID information for the specified RLPP must be set before setting the RLPF Target Port modifications. **Set** saves the changes made. **Reset** restores the RLPP to its previously saved values, eliminating all changes made to the VCIDs since the previous save. **Prev** refreshes the section with the previous item in the list. **Next** refreshes the section with the next item in the list. The VCID section is refreshed with the first APID in the new RLPP.

3.2.2.10.3 Forward Link Processing Function

The Forward Link Processing Function panel is sensitized when the option of the same name is set.

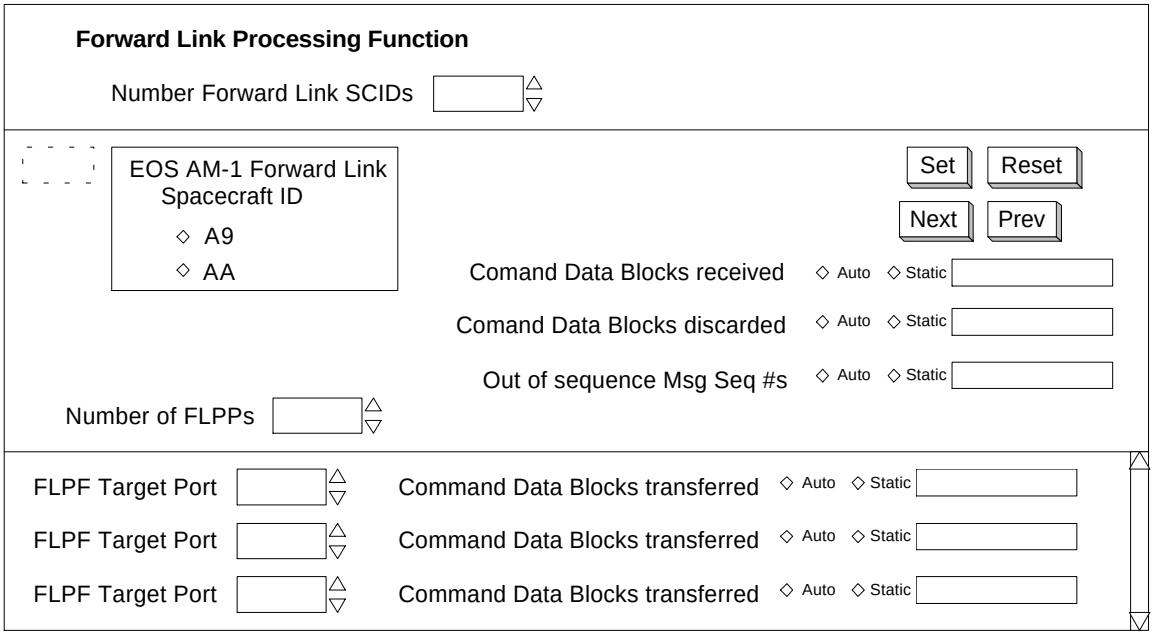

Figure 3-13. Forward Link Processing Function

The number of Forward Link SCIDs determines how many Forward Link SCIDs are in the CODA Report. Its range is 0 to 2. The ordinal number of the current SCID is displayed for information purposes in the field to the left of its identification. The ordinal number cannot be edited. The SCID identification can be modified by editing the field or clicking on the spinner arrows.

The number of FLPPs determines the number of FLPP items there are. The RLPF Target Port (RLPP) identifier field is changed by editing or clicking on its spinner arrows.

After entering all FLPP information for the specified SCID, click on the **Set** button for the SCID. **Set** saves the changes made. **Reset** restores the SCID to its previously saved values, eliminating all changes made to the FLPF Target Ports since the previous save.

3.2.2.10.4 Return Link Service Status

The Return Link Service Status panel is sensitized when the option of the same name is set.

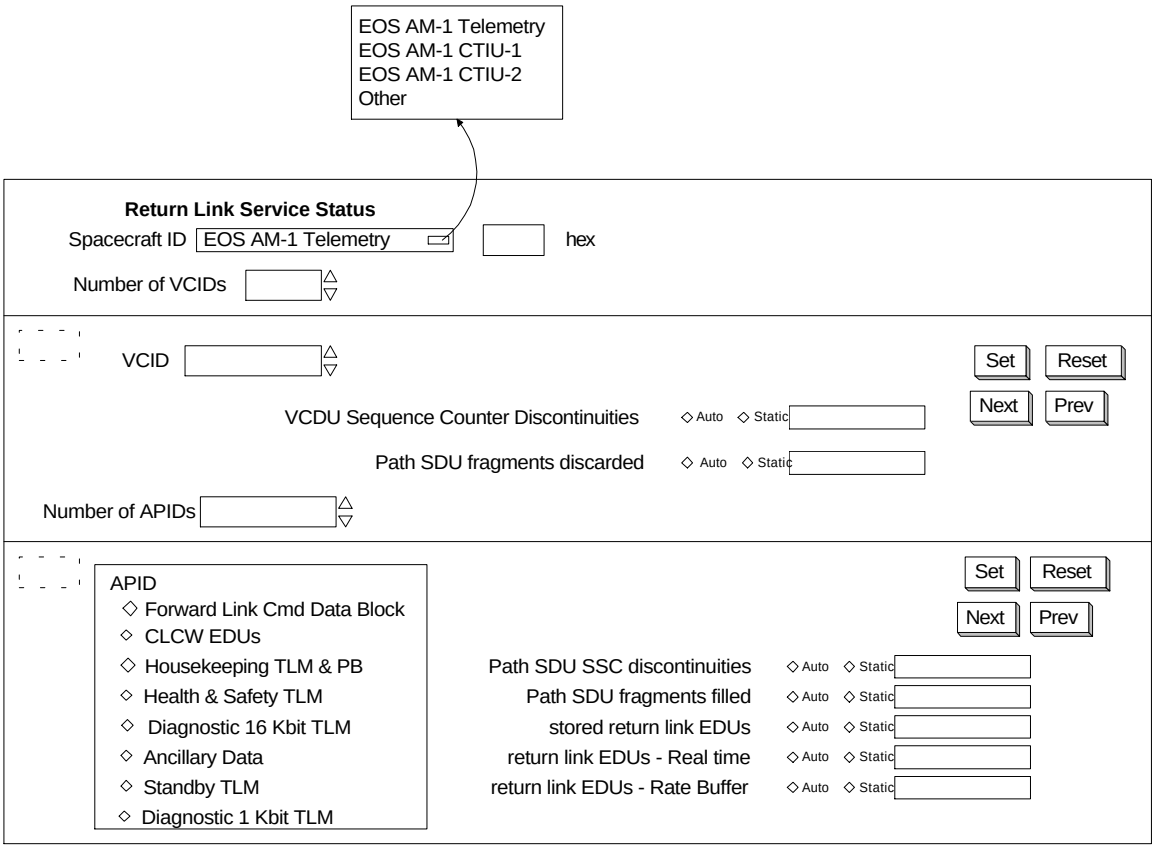

Figure 3-14. Return Link Service Status

The Spacecraft Id field is editable; a hexadecimal value less than 0xff is required.

The number of VCIDs determines how many VCIDs are found in the return link service status part of the OMD. The range is 0 to 255.

The number of APIDs determines how many APIDs are found in the current VCID. The range is 0 to 2047.

Each of the VCIDs and APIDs has an ordinal number field to the left of their identifiers. The ordinal number indicates which element in the list the current item is; it cannot be modified. The identification of both the VCID and APID can be modified.

The APID for the Return Link Service Status is a combination of the normal APID and the Spacecraft ID specified. The combination is made in the software.

Clicking on **Set** in the APID section saves the modifications made for the specified APID only. Clicking on **Reset** restores the previously saved APID parameters. The **Prev** and **Next** buttons refresh the APID section with the previous and next APID in the list.

Clicking on **Set** in the VCID section saves the modifications made for the specified VCID including all the APIDs modified. All APIDs for the specified VCID must be set before setting the VCID. Clicking on **Reset** restores the previously saved VCID parameters losing any changes made in the APID section. Prev and Next refreshes the VCID section with the previous and next VCID in the list. The APID section is refreshed with the first APID in the new VCID.

3.2.2.10.5 Forward Link Service Status

There is no information required from the operator for the Forward Link Service Status section of the CODA.

3.2.2.10.6 Ground Communications Return Link

The Ground Communications Return Link panel is sensitized when the option of the same name is set.

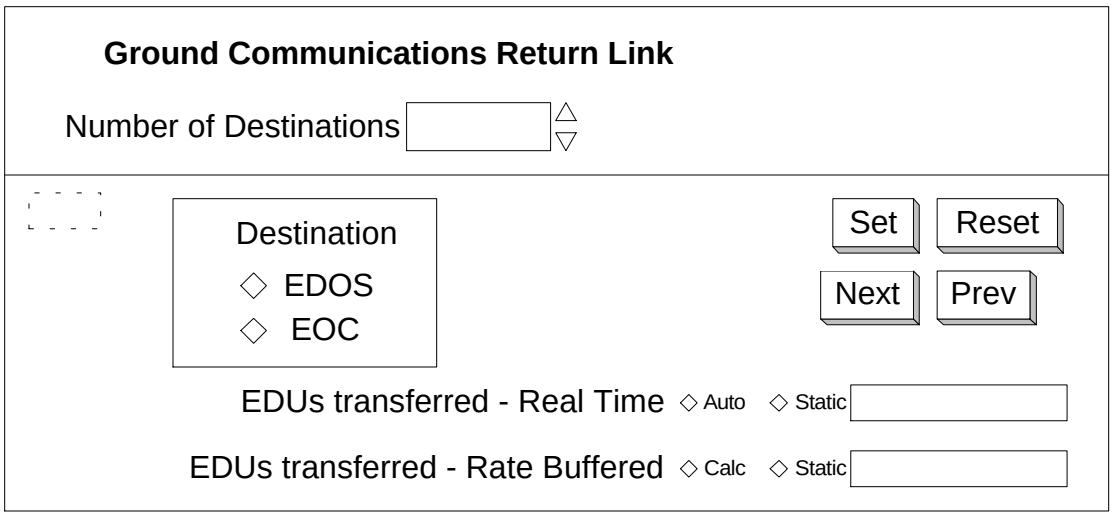

Figure 3-15. Ground Communications Return Link

The value in Number of Destinations field determines how many destinations are found in the ground communications return link part of the current OMD.

The ordinal number of the current destination is displayed for information purposes to the left of the destination identification. The identification can be modified, the ordinal number cannot be modified. The **Set** button sets the modifications made for the current destination. The **Reset** button restores the previously saved information for the current destination. **Prev** and **Next** refresh the destination section with the previous and next destination in the list.

3.2.2.10.7 Accounting Information Received

The Accounting Information Received panel is sensitized when the option of the same name is set.

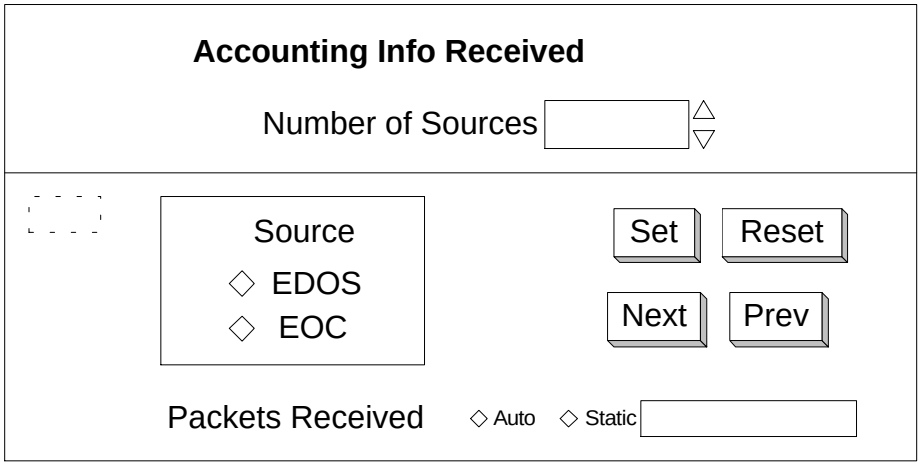

Figure 3-16. Accounting Information Received

The number of Sources determines how many sources are found in the accounting information received section. The range is 0 to 255.

The ordinal number of the current source is displayed for information purposes to the left of the source identification. The identification can be modified, the ordinal number cannot be modified. The **Set** button sets the modifications made for the current source. The **Reset** button restores the previously saved information for the current source. **Prev** and **Next** refresh the source section with the previous and next source in the list.

3.2.2.10.8 Accounting Information Transmitted

The Accounting Information Transmitted panel is sensitized when the option of the same name is set.

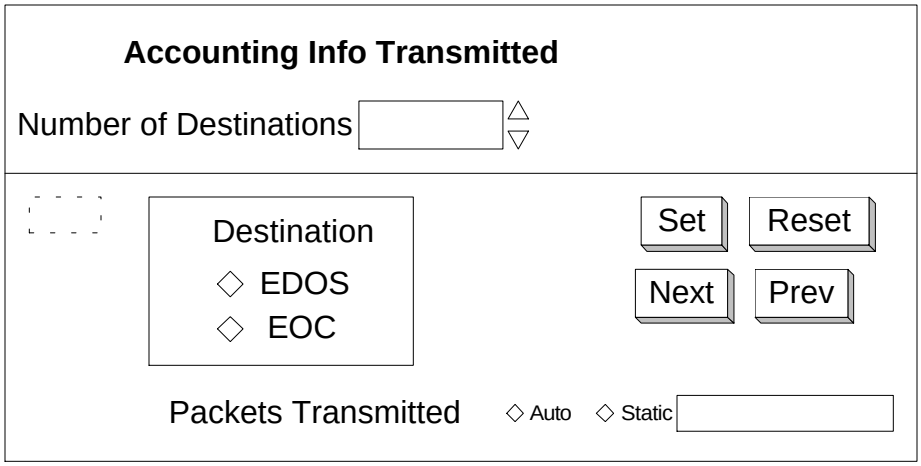

Figure 3-17. Accounting Information Transmitted

The number of Destinations determines how many destination sections there are. The range is 0 to 255.

The ordinal number of the destination is displayed to the left of its identification. The ordinal number indicates the relative position in a list and it cannot be modified. The identification can be modified.

Clicking on **Set** saves the parameters for the specified Destination ID. **Reset** restores the previously saved parameters for the specified Destination ID. Prev and Next refresh the destination section with the previous and next destination in the list

3.2.2.11 EDS/PDS Delivery Request Editor Panels

Figures 3.18 through 3.20 depict the OMD Editor panels for the EDS/PDS Delivery Request OMD. Like the CODA Report panels, these panels contain **Set**, **Reset**, **Next**, and **Prev** buttons for subsections and *Auto* and *Static* options for count parameters. These features are operated and behave in the same way for both types of OMD.

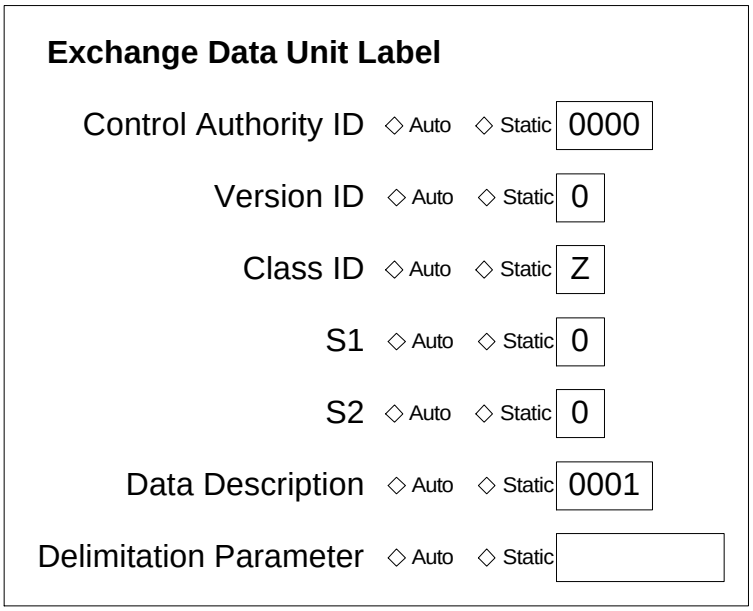

Figure 3-18. Exchange Data Unit Label

PDS / EDS Delivery Record Label

Control Authority ID	◇ Auto	◇ Static	0000
Version ID	◇ Auto	◇ Static	0
Class ID	◇ Auto	◇ Static	Z
S1	◇ Auto	◇ Static	0
S2	◇ Auto	◇ Static	0
Data Description	◇ Auto	◇ Static	0001
Delimitation Parameter	◇ Auto	◇ Static	

Figure 3-19. PDS/EDS Delivery Record Label

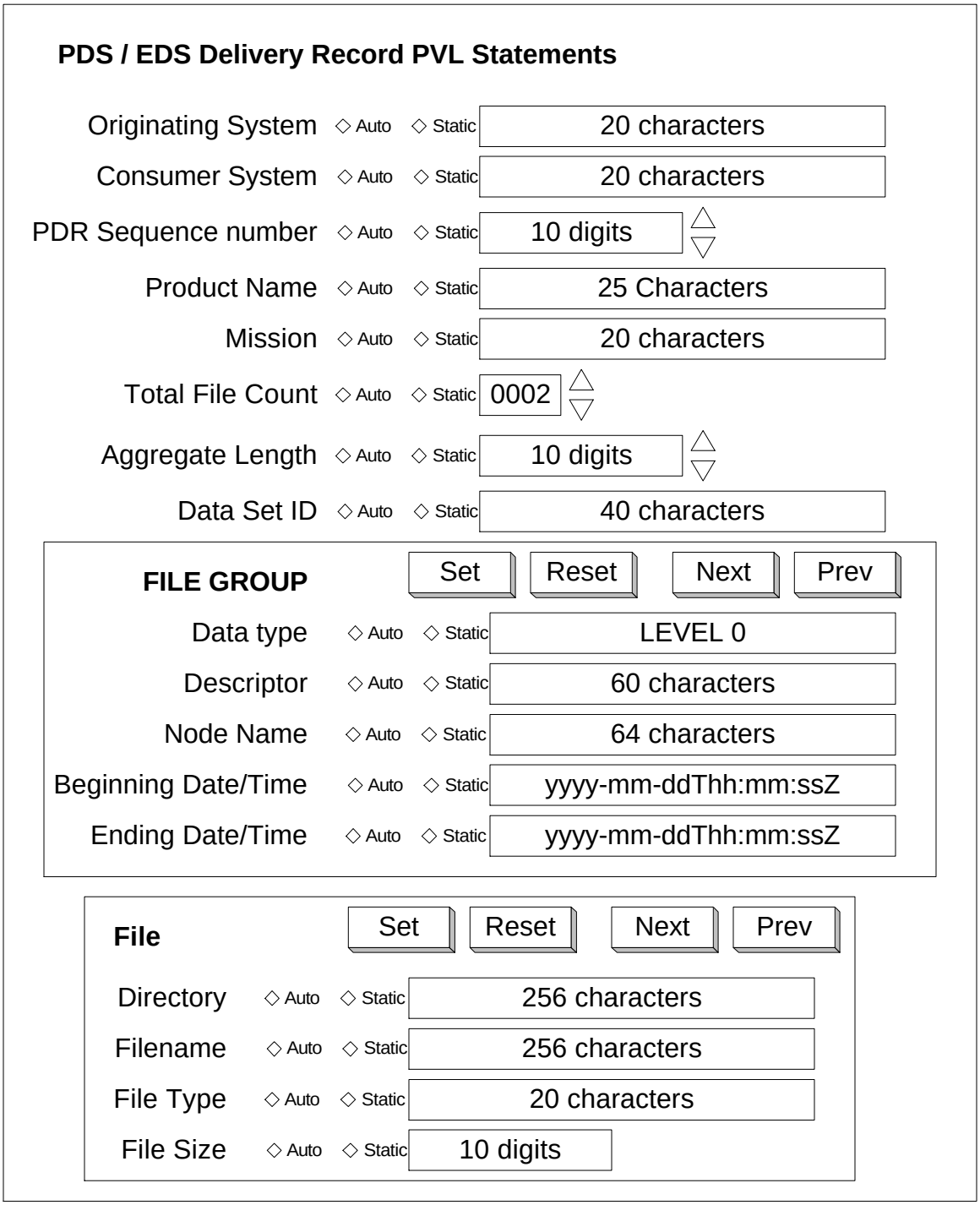

Figure 3-20. PDS/EDS Delivery Record PVL Statements

3.2.2.12 Ground Message Header

The Ground Message Header dialog box, shown in Figure 3-21, appears when the operator selects *Header...* under Edit in the OMD Editor or if the OMD is new as specified under File in the main window or the **Set** button in the "New Record" section of the Scenario Editor.

The dialog box is initialized with header values in the existing OMD or from the default OMD file if the OMD is new.

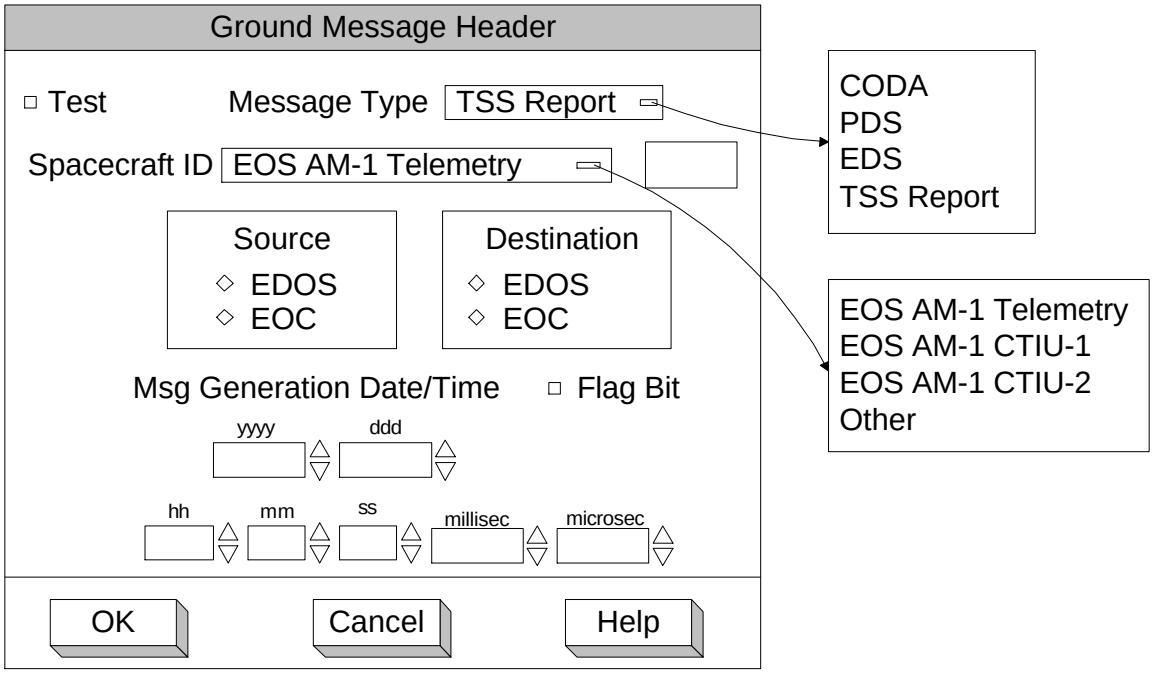

Figure 3-21. Ground Message Header Dialog Box

The **Test** button is set by default. When it is set, 128 is added to the message type specified in the Message Type option menu. The OMD is always transmitted as the type that is edited. The message type specified in the header may not match the type of OMD being edited. Selecting a type from the Message Type option menu changes the message type written in the ground message header.

Selecting *Other* from the Spacecraft ID option menu causes the field to its right to be sensitized. This field is editable and accepts a 2 digit hexadecimal value to replace the "normal" spacecraft id.

The Message Generation Date and Time are set as in the CODA Report Content Indicator panel described above. If Flag Bit is set, the software will reference PB-5 Time code "option C".

Clicking on **OK** saves the header with the current OMD and closes the dialog box. **Cancel** only closes the Ground Message Header dialog box. If the header was not set, a question box appears with the message "Changes have been made to the current header. Save the changes with the current OMD?". Clicking on **Yes** sets the header and continues closing the dialog box. Clicking

on **No** closes the dialog box without setting anything; changes made are lost. Clicking on **Cancel** returns control to the Ground Message Header dialog box.

3.2.2.13 OMD Traffic

The OMD Traffic dialog box, shown in Figure 3-22, appears when the operator clicks on *OMD Traffic...* under View in the main window.

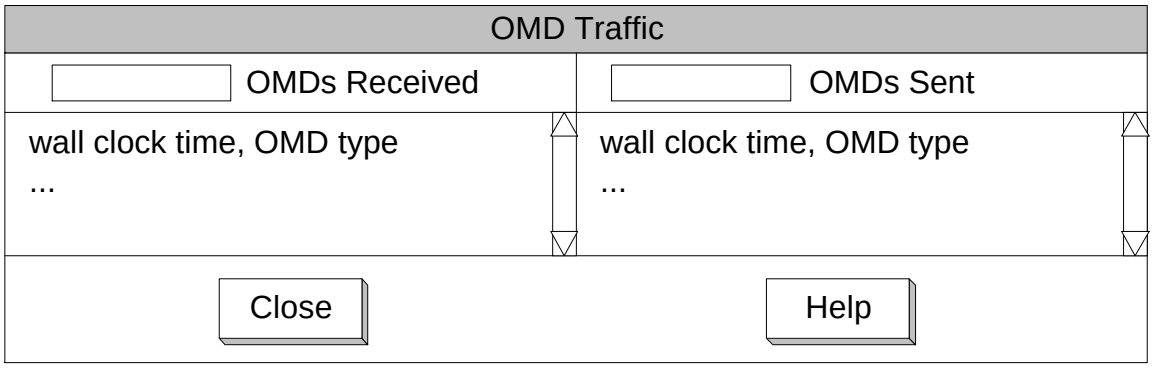

Figure 3-22. OMD Traffic Dialog Box

This dialog box displays the number of OMDs Received and Sent and the time and type of each.

Clicking on **Close** closes the window.

3.2.3 Processing

The OMD Simulator is comprised of a main module and six subprocesses. The main module, om_main, contains the user interface and spawns the subprocesses to execute asynchronously. Communication between the subprocesses is implemented using fifos (named pipes). To distinguish operation with the HRS, LRS and MPS, a command line argument will be used to sensitize or desensitize user interface features unique for each configuration.

Figure 3-23 shows the structure chart of om_main, the highest level process of OMDSIM. Section 3.2.4 contains a detailed description of each of the following six subprocesses, including prolog, PDL, and structure chart:

- om_scenario_manager, which uses information provided by the operator to execute the preestablished transmission schedule of OMD to other EOSDIS test elements.
- om_scheduler, which causes the execution of the OMD transmission schedule to occur on time for each element of the schedule.
- om_transmission_task, which merges OMD information from the user, from processes external to OMDSIM (i.e. the LRS status buffer interfacing software) and (optionally) counters evaluated just prior to transmission into final OMD, converts the information into a form suitable for transmission, and finally, transmit the information to other EOSDIS test elements.

- om_logger, which creates and maintains the session event and OMD log files.
- om_reception_task, which accepts OMD from other EOSDIS test elements.
- om_delogger, which views the logged OMD records.

In addition to these subprocesses, om_main is supported by several functions to implement the design for the graphical user interface (GUI). The diagrams and text in Section 3.2.2 serve as the PDL for the GUI. In Figure 3-23, OMDSIM callback functions are referred to as "callbacks". A brief overview of the more important OMDSIM routines is given in Table 3-2.

Table 3-2. OMDSIM Software Routines

Name	Description
om_init_gui	perform GUI initialization
om_get_default_OMD	supports om_init_gui by providing the interface to default OMD values
om_create_gui	creates the realtime display of the GUI
ui_establish_comm_link_CB	(callback function) support for all external communication interfaces for OMDSIM
om_establish_comm_link	all subsystem dependent support for external communication for OMDSIM
op_input_communication	all callback routines required for operator input, including scenario file I/O
om_spawn_subs	activation of all subprocesses

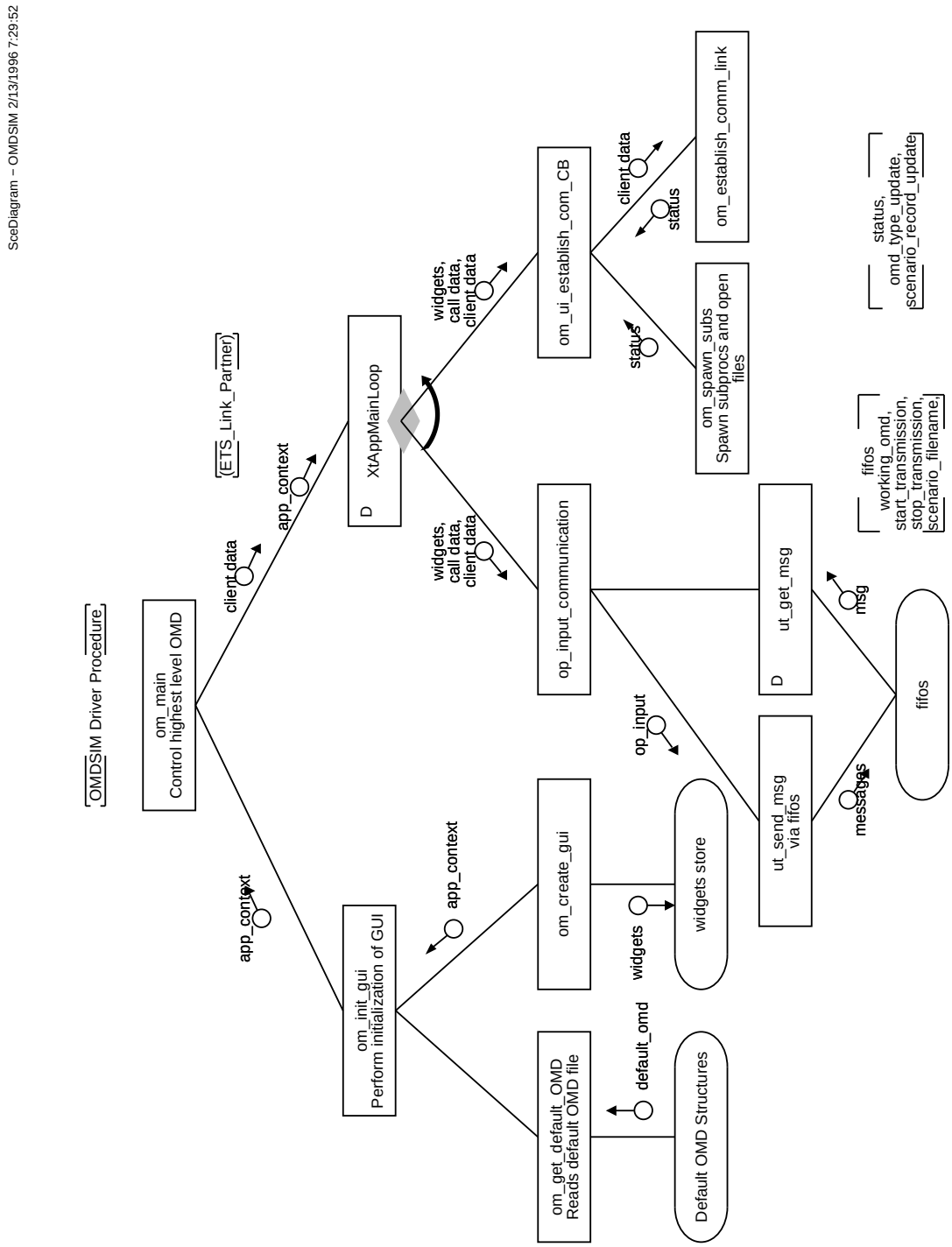

Figure 3-23. Structure Chart for OM_Main

3.2.4 Prologs/Program Design Language/Structure Charts for Module Drivers

The following sections present the prolog, PDL, and structure charts for the major subprocesses of OMDSIM.

3.2.4.1 om_scenario_manager

Purpose:

The om_scenario_manager subprocess is solely responsible for the maintenance, execution, interruption and termination of the OMD transmission schedule, known as the scenario. It reads the store of OMD and makes pregenerated OMD available for transmission. The structure chart for the om_scenario_manager subprocess is shown in Figure 3-24.

External References:

om_sm_initialize	initialize scenario manager input and output fifos
fopen, fscan	standard IO utilities
ut_send_msg	send to output queue
ut_get_msg	fetch from input queue

Input:

Scenario_file_name	the name of the file containing the transmission scenario to be loaded into the system.
Stop_transmission	end of transmission requested
ready_for_next_record	normal end of scenario step

Output:

working_OMD	OMD bit stream and source information
transmission_interval	interval between OMD transmissions
transmission_duration	duration of OMD transmission

Notes:

The om_scenario_manager communicates this scheduling information (the duration of the sequence and interval between transmissions of one kind of OMD) to the om_scheduler subprocess using Fifos.

The om_scenario_manager waits for completion of this portion of the scenario until a signal is received from the om_scheduler subprocess indicating that processing may continue to the next portion of the scenario.

Requirements Met:

V1-1.1.3.1-1; V1-1.1.3.8-2

PDL:

```
use ut_get_msg to obtain filename from UI
use fopen to open the scenario file
until EOF or stop_transmission received loop
        use fscan to read one scenario record
        uses ut_send_msg to put working_OMD to the fifo
        use fscan to get transmission_interval
        use fscan to get transmission_duration
        use ut_send_msg to send transmission_interval, transmission_duration to the fifo
        wait for ready_for_next_scenario_record signal (from om_scheduler)
end until
close file
if stop_transmission not received
        send stop_transmission to fifo using om_send_msg
end if
```

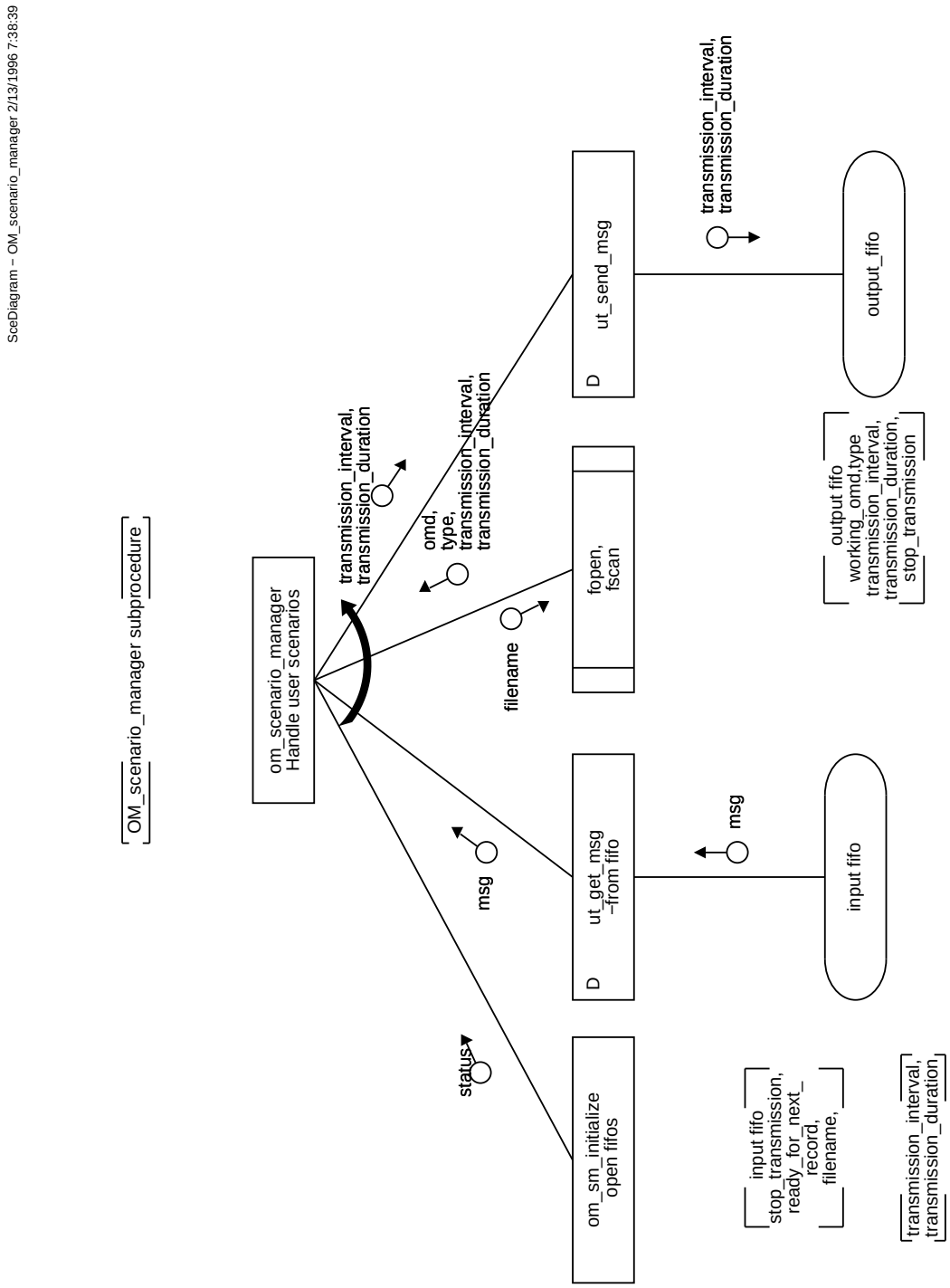

Figure 3-24. Structure Chart for OM_Scenario_Manager

3.2.4.2 om_scheduler

Purpose:

The om_scheduler subprocess sends transmit_OMD signals to the om_transmission_task subprocess via the fifo so that the OMD are transmitted according to the scenario or schedule specified. The structure chart for the om_scheduler subprocess is shown in Figure 3-25.

External References:

ut_get_msg	gets the current OMD transmission schedule information from the input fifo
om_sc_initialize	open fifos
ut_get_msg_from_socket	LRS interface routines
ut_send_msg	sends start/stop transmission signal to output fifo

Input:

Start_transmission

Stop_transmission

transmission_interval

transmission_duration

Output:

ready_for_next_scenario_record (signal)

transmit_OMD (signal)

stop_transmission

Notes:

Separating the functions of the om_scenario subprocess from the om_scheduler subprocess allows the operator to generate OMD “on the fly”, without requiring a scenario file to exist.

Requirements Met:

V1-1.1.3.1-1; V1-1.1.3-2

PDL:

```
use om_sc_initialize to open fifo
until stop_transmission received (from UI or LRS or MPS or om_scenario_manager)
        get current scheduled transmission_interval
        get current transmission_duration
        calculate the remainder of the interval (using the operating system clock)
        wait for the remainder of the interval designated between executions
        check for stop_transmission
        if stop_transmission not received
                send the transmit_OMD (signal) to the output fifo
        end if
end loop
send stop (signal) to terminate the output fifo
```

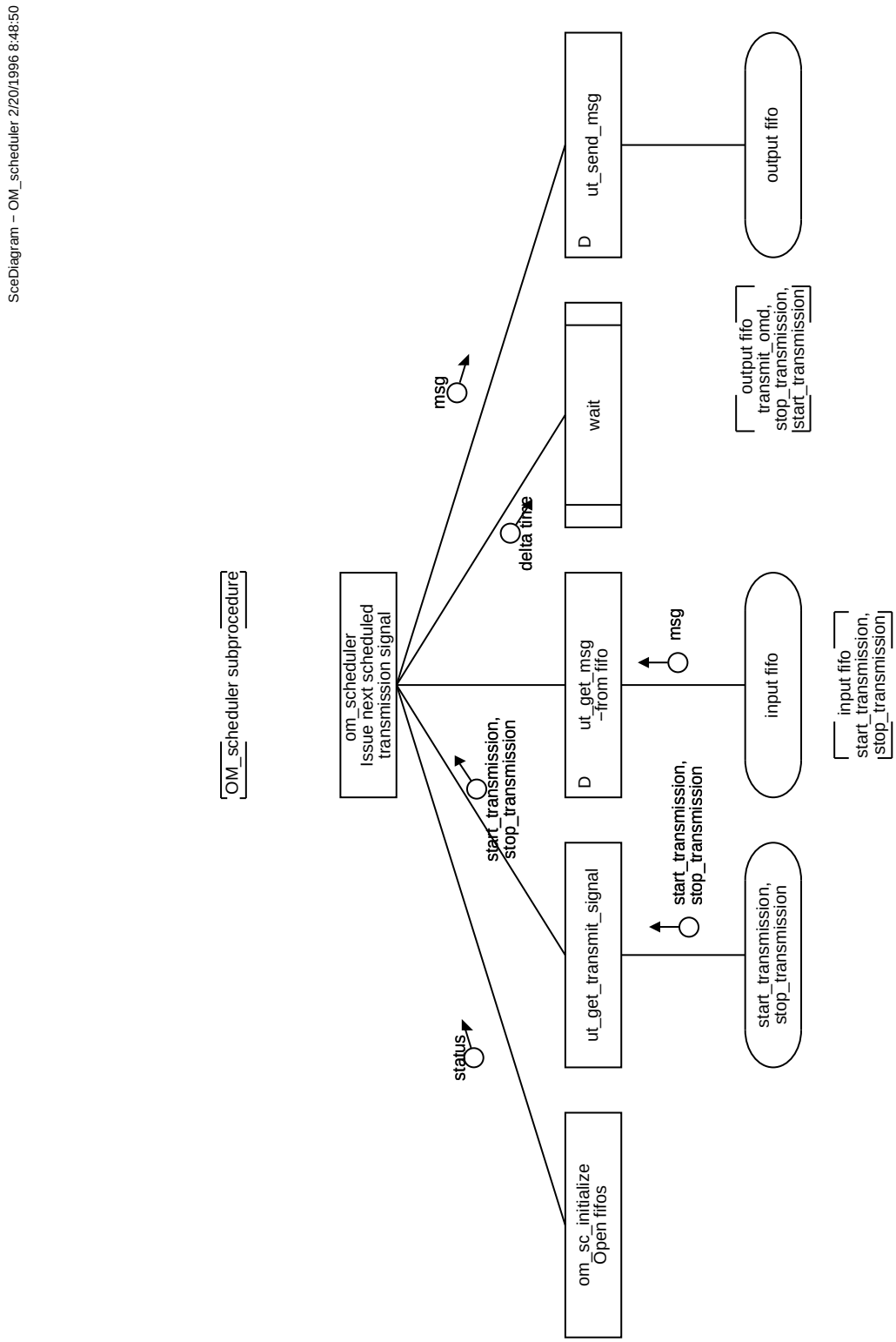

Figure 3-25. Structure Chart for OM_Scheduler

3.2.4.3 om_transmission_task

Purpose:

The om_transmission_task merges OMD elements from three possible sources (static/default values, calculation as directed by the working_OMD structure, or the hardware environment). After the OMD input information has been merged, om_transmission_task converts it to a bit string and sends the bit string to other EOSDIS test elements via EBnet. The structure chart for the om_transmission_task subprocess is shown in Figure 3-26.

External References:

om_tt_initialize	open fifos
ut_get_msg	fetch working_OMD and stop_transmission signal from fifo
om_send_to_external	any installation specific interface procedures to transmit via EBnet.
om_calculate_element	update time-dependent one-up counters
om_tt_merge	merge OMD elements from hardware, calculation and working OMD
ut_get_from_partner	socket connection to LRS/MPS for OMD elements
ut_send_msg	output final OMD to fifo for logging.

Input:

transmit_OMD

stop_transmission

working_OMD

omd_elements

Output:

OMD (transmitted in bit form)

accounting_statistics

Notes:

By separating the om_scheduler functions from the om_transmission_task functions, the transmission task is able to receive signals from the operator and from om_scheduler subtask independently, allowing the scenario file to act as separate source of OMD information.

Requirements Met:

V1-1.1.3.1-1; V1-1.1.3.4-1; V1.1.1.3.4-3; V1-1.1.3.5-1; V1-1.1.3.8-1; V1-1.1.3.8-2; V1-1.1.3-2

PDL:

```
initialize local variables
use om_tt_initialize to open fifos
until stop-signal received (from om_scheduler) via fifo
        get working_OMD (from fifo)
        if working_OMD.OMD_type == CODA then
                call om_tt_merge (CODA)
        elseif working_OMD.OMD_type == EDS/PDS_delivery_record then
                call om_tt_merge (EDS/PDS)
        use ut_send_to_external ((bit_string) OMD) to send the final OMD via EBnet

        elseif working_OMD.OMD_type == other then
                processing for newly implemented OMD types, if realtime information
                is required
        end if
        update accounting statistics for transmitted OMD
        use om_logger ((bit_string)OMD) to log the OMD
        check for stop_transmission
end until;
convert TSS to bit string
use om_send_to_external ((bit_string) OMD) to send the final OMD via EBnet
```

```
use om_logger ((bit_string)TSS) to log the TSS
```

The subfunction om_tt_merge (CODA) is as follows:

```
for each CODA one-up counter do
        if working_OMD.assoc_data_source == LRS_Status then
                request and get OMD_element from the
                External_OMD_queue
        elseif working_OMD.assoc_data_source = Auto then
                use om_calculate_element (OMD_element_rate_data) to
                update the one-up counter
        elseif working_OMD.assoc_data_source = static then
                null;
        end if;
end do-for
convert final structure to bit string
```

The subfunction om_tt_merge (EDS/PDS) is as follows:

```
        for each EDS/PDS one-up counter do
                if working_OMD.assoc_data_source == LRS_Status then
                        request and get OMD_element from the
                        External_OMD_queue
                elseif working_OMD.assoc_data_source = Auto then
                        use om_calculate_element (OMD_element_rate_data) to
                        update the one-up counter
                elseif working_OMD.assoc_data_source = static then
                        null;
                end if;
        end do-for
        convert final structure to bit string
```

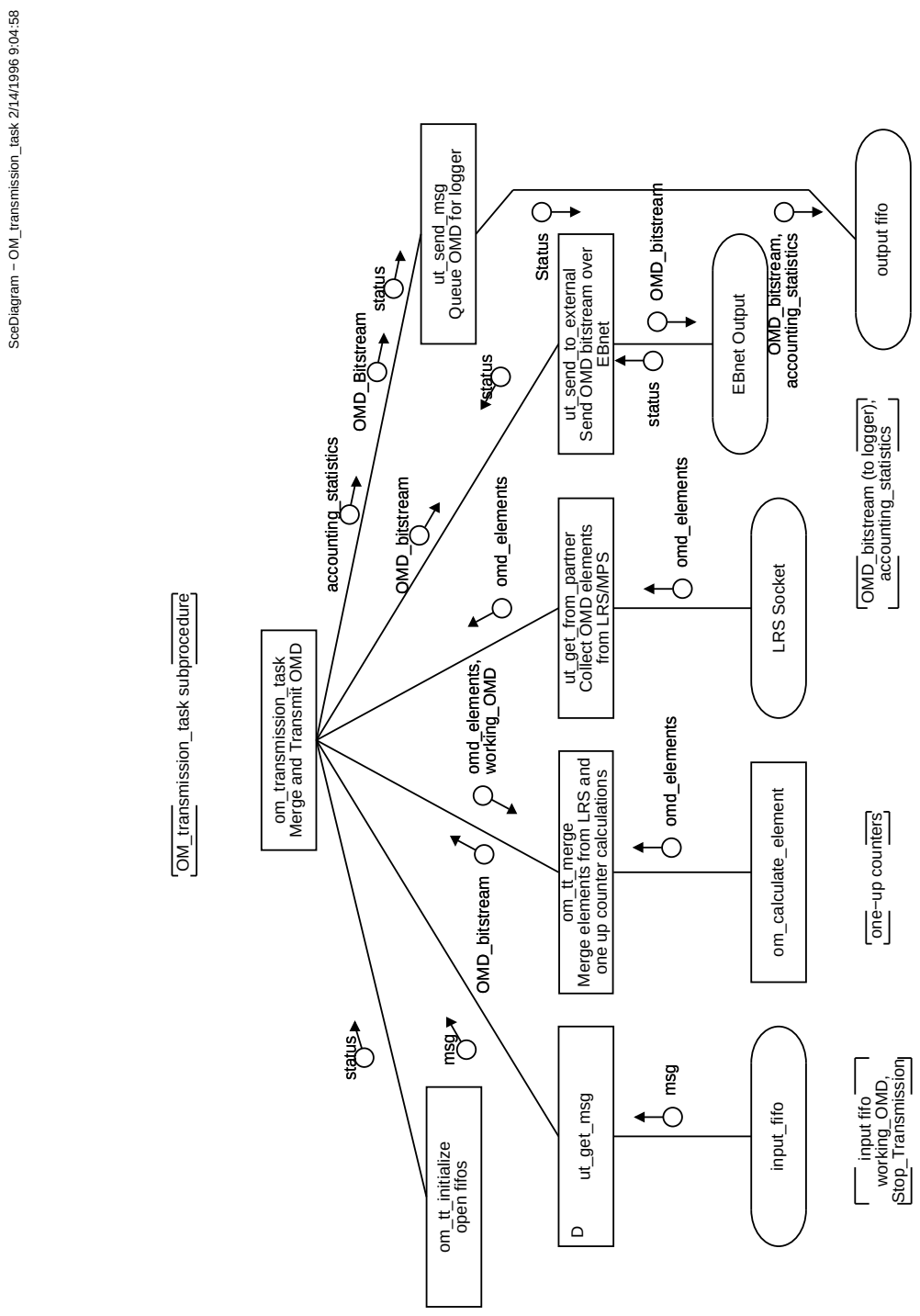

Figure 3-26. Structure Chart for OM_Transmission_Task

3.2.4.4 om_logger

Purpose:

The om_logger subprocess is responsible for the maintenance of the OMDSIM event log and of the logs of OMD transmitted and received. The structure chart for the om_logger subprocess is shown in Figure 3-27.

External Reference:

om_lo_initialize	open fifo
ut_get_msg	get OMD_bitstream, start/stop transmission from fifo
om_log_write_entry	disk status and write routines

Input:

log_data_type	indication of type of data to be logged
accounting_statistics	number of received/transmitted OMD
stop_transmission	stop transmission signal

Output:

OMD_Received log file

OMD_Transmitted log file

OMD event file

Requirements Met:

V1-1.1.3.4-2; V1-1.1.3.4-3; V1-1.1.3.6-1; V1-1.1.3.6-2

PDL:

create the name of the OMD-received log file

create the name of the OMD-transmitted log file

create the name of the OMD-event log file

open log files

```
until stop_transmission signal
        get log_data_type and data to be logged
        get current time
        if log_data_type = OMD-received then
                write OMD, current time to OMD-received log file
        elseif log_data_type = OMD-transmitted then
                write OMD, current time to OMD-transmitted log file
        elseif log_data_type = event
                write ASCII text string, current time to OMD-event log file
        end if
end until
use ut_get_msg to get accounting statistics from Fifos
log accounting statistics
close the files
```

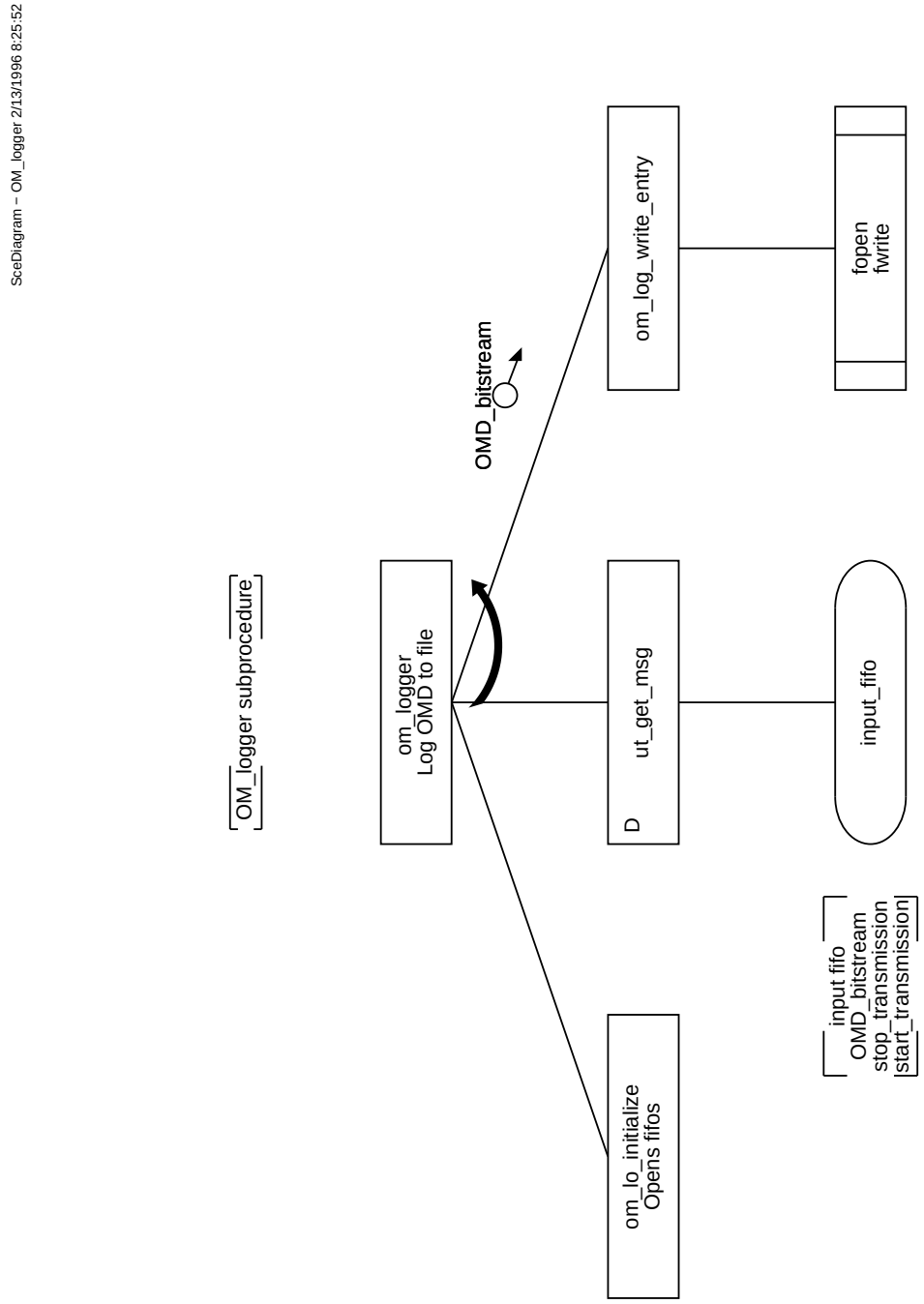

Figure 3-27. Structure Chart for OM_Logger

3.2.4.5 om_reception_task

Purpose:

The om_reception_task subprocess accepts and logs (via the om_logger subprocess) OMD received from other EOSDIS test components. The structure chart for the om_reception_task subprocess is shown in Figure 3-28.

External References:

om_re_initialize	open fifo
ut_get_over_EB	receive OMD from EBnet
ut_send_msg	write OMD/accounting statistics to fifo

Input:

OMD from other EOSDIS test elements

termination signal

Output:

accounting_statistics

log_data_type

OMD

Requirements Met:

V1-1.1.3-1; V1-1.1.3.2-1; V1-1.1.3.3-1; V1-1.1.3.4-2; V1-1.1.3.4-3

PDL:

Performs all handshaking procedures required by the interface prior to communication when spawned.

```
until termination-signal
        wait for OMD-received from EBnet
        update to the accounting statistics for incoming OMD
```

```
        any TBD actions bases on received OMD are performed
        use ut_send_msg to write OMD to fifo
end until
use ut_send_msg to log accounting statistics to the event log via fifo
```

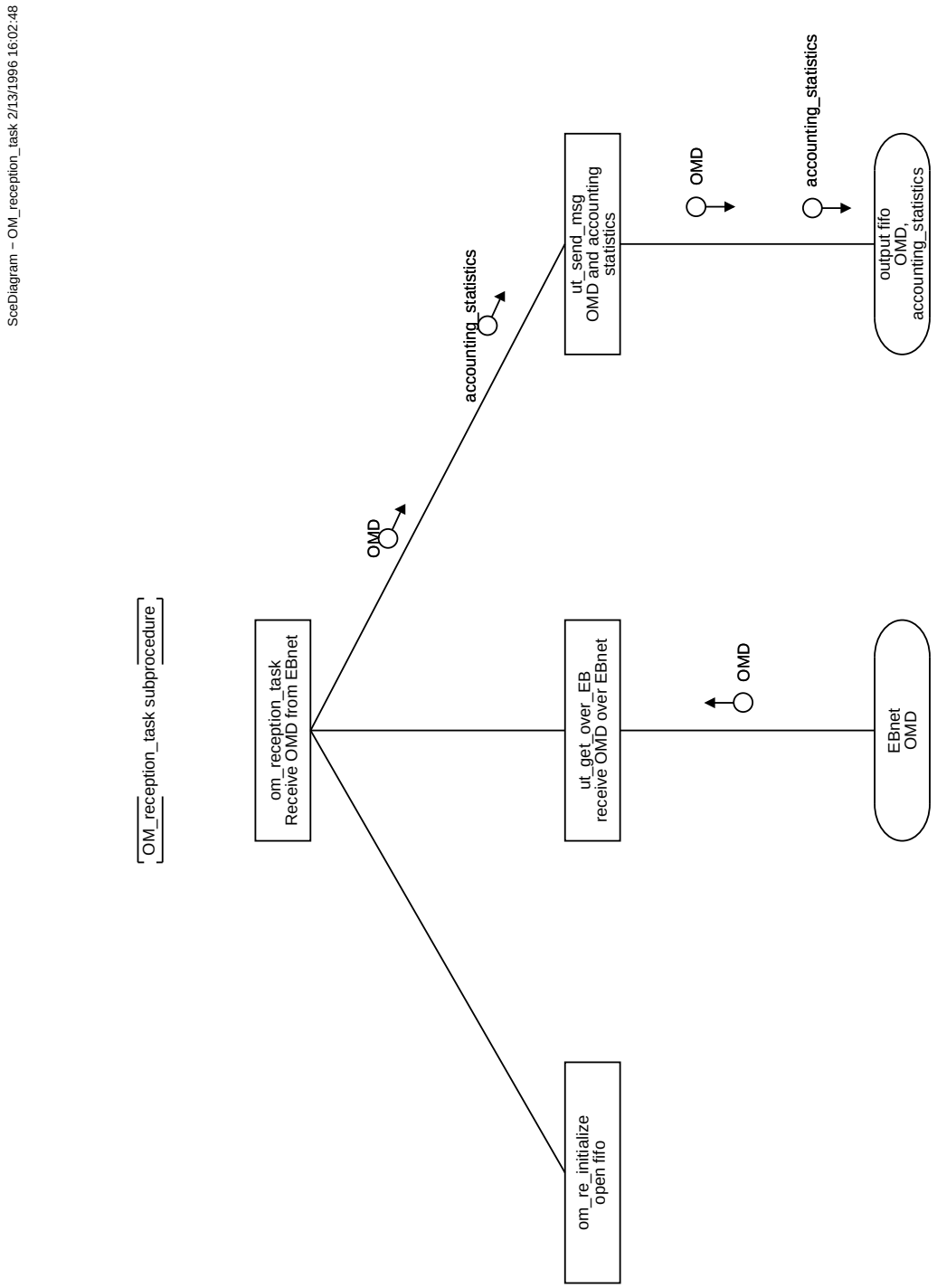

Figure 3-28. Structure Chart for OM_Reception_Task

3.2.4.6 om_delogger

Purpose:

The om_delogger subprocess is executed post test to view the logged OMD. The structure chart for the om_delogger subprocess is shown in Figure 3-29.

External References:

none

Input:

OMD_Log file name

(logged) OMD from OMD_Log

Output:

none

Requirements Met:

V1-1.1.3.7-1

PDL:

```
Open OMD_Log
until EOF loop
        get user signal to continue
        get OMD for GUI
end until
close OMD_Log
```

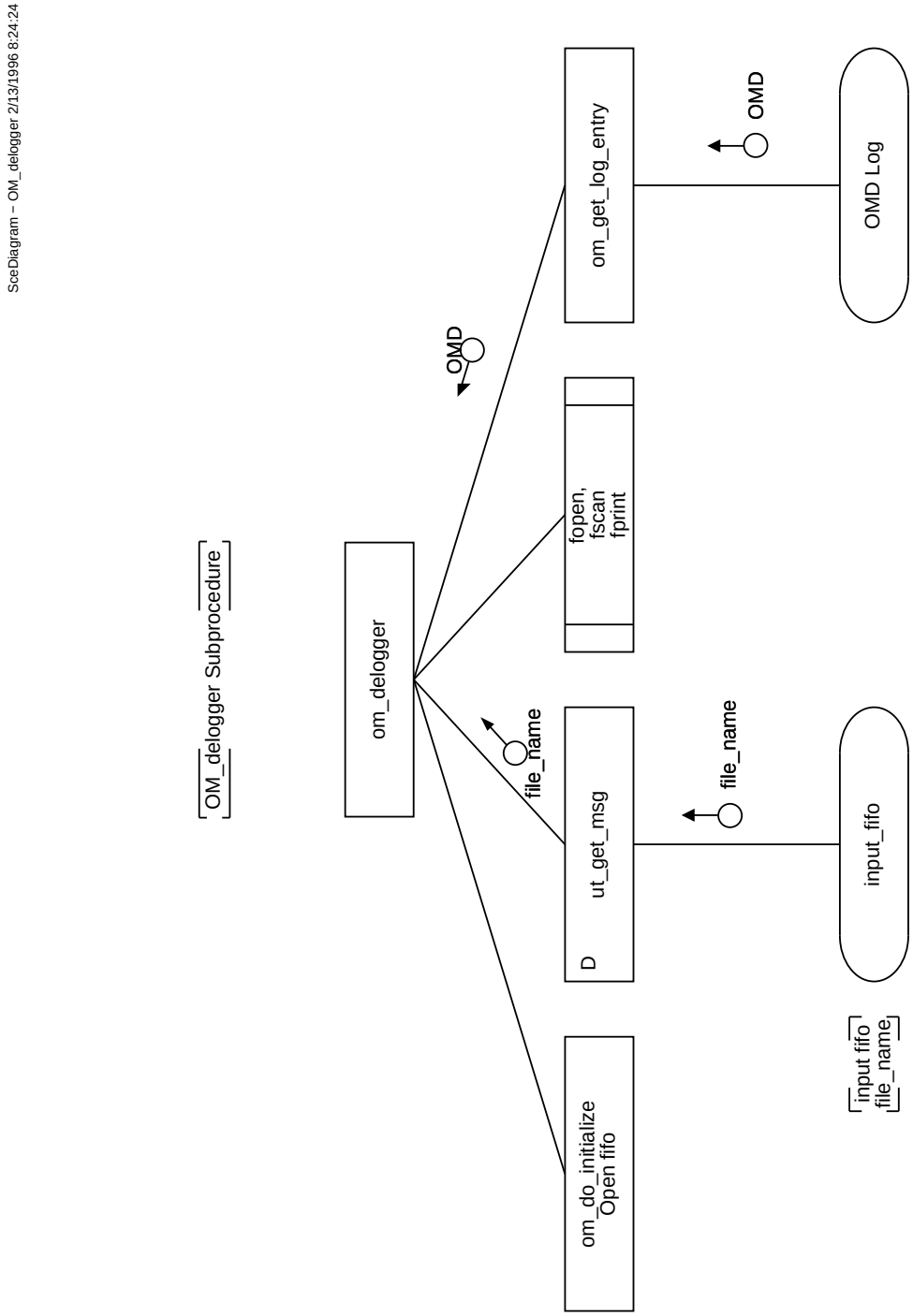

Figure 3-29. Structure Chart for OM_Delogger

3.3 Rate Buffered File Utility

3.3.1 Description

The Rate Buffered File Utility will be present only on the Low-Rate System. In order to support ETS test configuration 5, EDUs from the VLS will be sent to the CDS. The Rate Buffered File Utility will write the EDUs to a file. On operator request the rate buffered file is transferred to the EOC using KFTP. The file naming conventions and formats for rate buffered files are defined in the ICD between EDOS and the EOC (Reference 10).

The Rate Buffered File Utility is made up of two software processes. One task, rb_main, controls the user interface and the other task, rb_rbuf_main, performs the rate buffering function. The rate buffering task is invoked as part of LRS TPCE activation. The user interface task is activated and deactivated as desired by the operator to access rate buffering information.

3.3.2 User Interface Design

When the *Rate Buffered File Utility* is selected from an LRS TPCE menu, its main window appears as illustrated in Figure 3-30. The current rate buffered file name, file size and mode are shown.

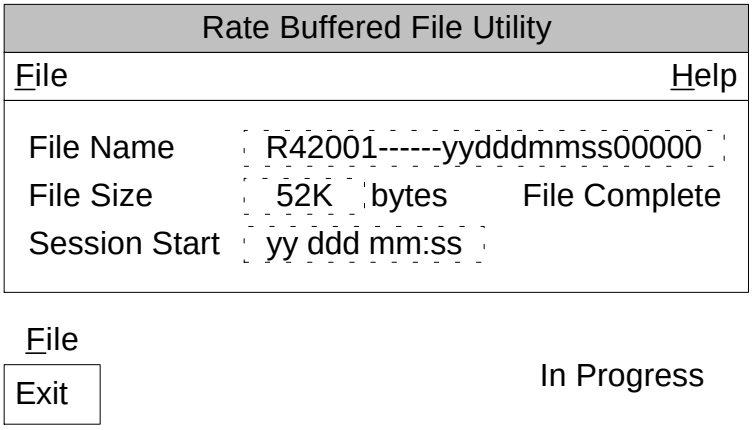

Figure 3-30. Main Window for Rate Buffering File Utility

The *File* menu provides the Exit function.

A file manager process, emulating that provided by HP VUE, will be available from the menu controller to provide file functions such as viewing, renaming, deleting and transferring of rate buffered files..

Major problems encountered during the processing of rate buffered data, such as file i/o errors or lack of rate buffered data for an entire session will be recorded in the LRS TPCE's system activity log and accessible through TPCE's main window.

3.3.3 Processing

The application will communicate with the TPCE ExternalStatusServer in order to receive notifications of event start and stops and to receive EDUs from the Low Rate hardware. The start and stop of data events will signal creation of new rate buffered files. Rate buffered files will be deleted from the system when requested by the operator or as additional space is needed.

3.3.4 Prologs/Program Design Language/Structure Charts for Module Drivers

The structure chart for the Rate Buffered Data Utility task is shown in Figure 3-31.

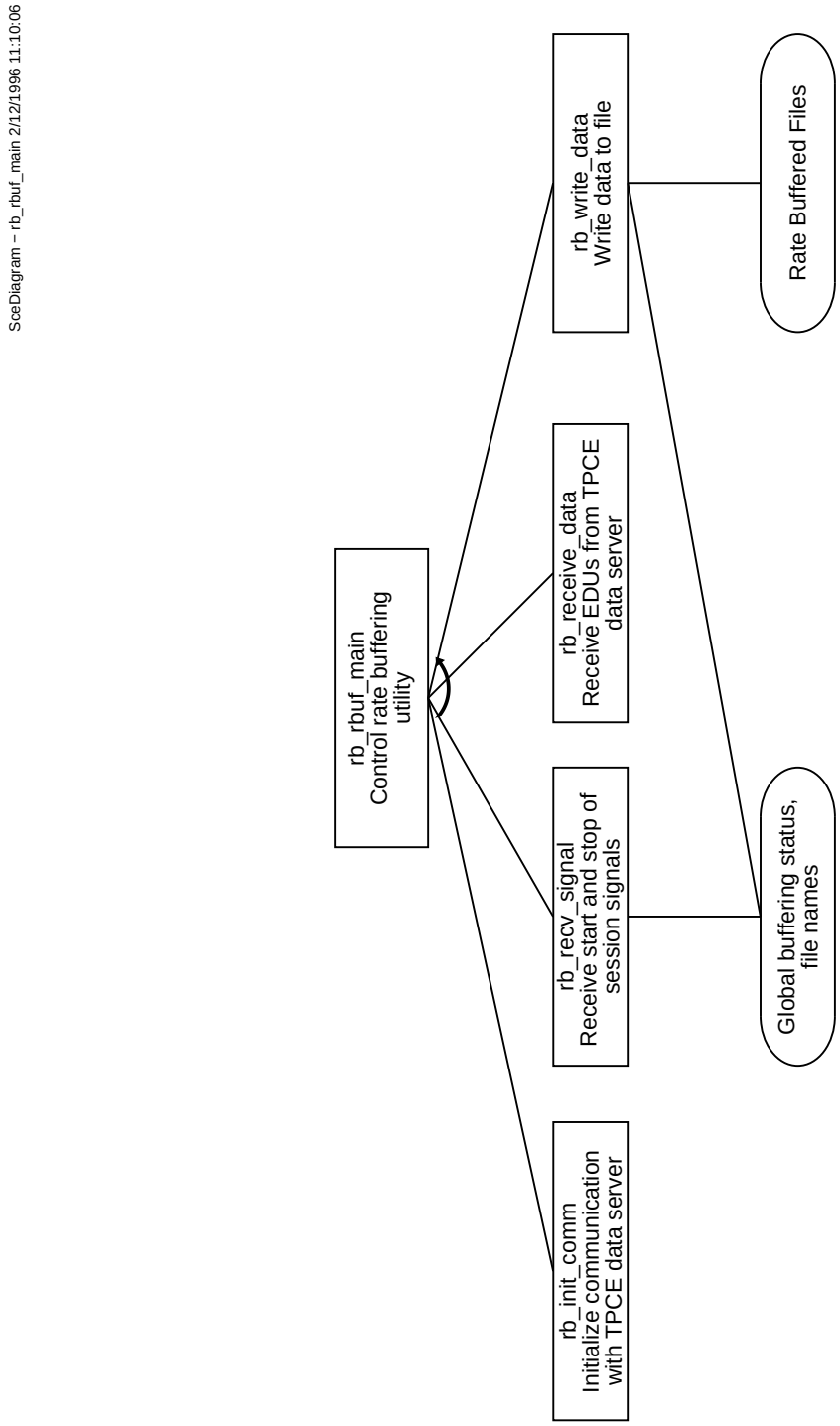

Figure 3-31. Structure Chart for Rate Buffering Process

3.4 Command Archive Utility

3.4.1 Description

The command archive utility is present only on the Low-Rate System. Whenever the Low Rate hardware receives command blocks, they are sent to the workstation for short term archiving. As a runtime utility, this utility is integrated into the LRS TPCE component.

The Command Archive Utility is made up of two software processes. One task, ca_main, controls the user interface and the other task, ca_archive_main, performs the archiving function. The command archiving task is invoked as part of LRS TPCE activation. The user interface task is activated and deactivated as desired by the operator to access command archive information.

3.4.2 User Interface Design

When the *Command Archive Utility* is selected from an LRS TPCE menu, its main window appears as illustrated in Figure 3-32. The current Command Archive File names, and file sizes are shown.

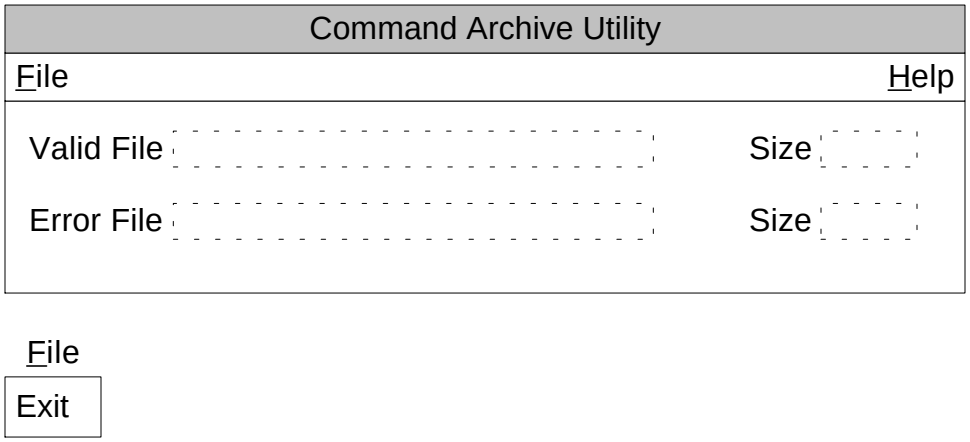

Figure 3-32. Main Window for Command Archive Utility

The *File* menu provides a *View...* function and an *Exit* function..

View... activates a file selection box. After a file is selected and the **OK** button is clicked, scrollable window is provided for the viewing of the file. A sample view window is shown in Figure 3-33.

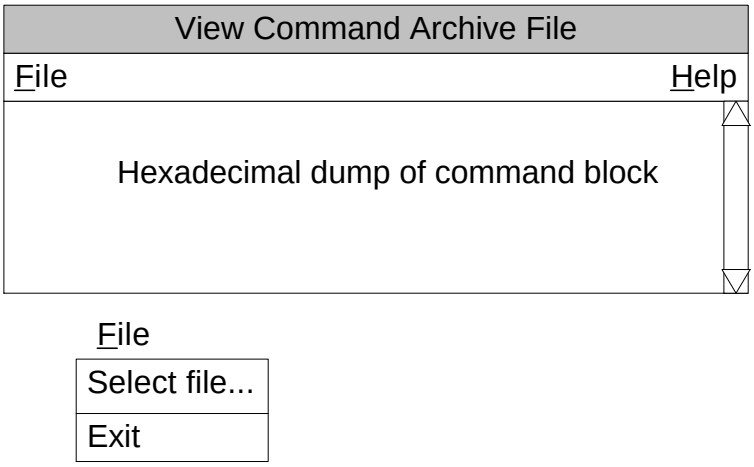

Figure 3-33. View Window for Command Archive Files

The *File* menu provides a *Select file...* function and an *Exit* function..

The *Select file...* option allows the operator to change the file being viewed.

Major errors encountered during the processing of command blocks, such as file i/o errors or lack of command blocks for an entire session will be recorded in the LRS TPCE's system activity log.

3.4.3 Processing

The command archive utility registers with the TPCE ExternalDataServer in order to receive event notification when sessions are started and stopped and to receive the command blocks from the VLS hardware. There are two socket connections, one for command blocks in error, the other is for all command blocks processed by the VLS. In addition to storing the blocks, the utility will provide rudimentary display and dump capabilities. Command archive files are available for transfer to other sites using KFTP. Command archive files are deleted from the system when requested by the operator or as additional space is needed.

3.4.4 Prologs/Program Design Language/Structure Charts for Module Drivers

The structure chart for the command archive utility is shown in Figure 3-34.

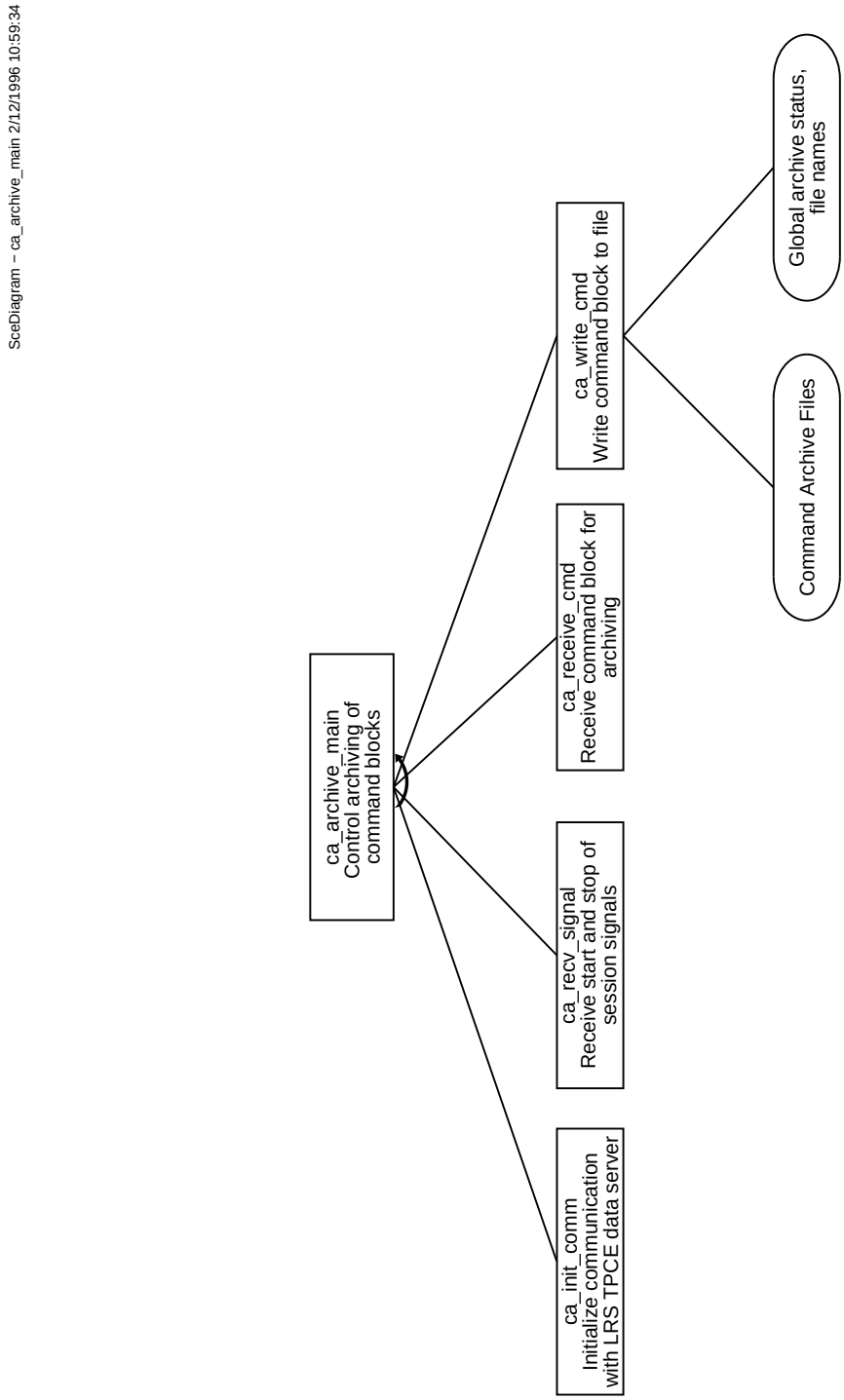

Figure 3-34. Structure Chart for Command Archive Process

3.5 Data Verification Tool

3.5.1 Description

The Data Verification Tool is present only on the High-Rate System. This tool verifies that fields in the Construction Record (CR), primary and secondary packet headers for a specified PDS or EDS comply with valid format and range of values as specified in the EDOS-EGS ICD (Reference 10). For some fields, an additional verification is made between the CR value and the value found in the packet headers.

The DVT is made up of two software processes. One task, dv_gui_main, controls the user interface and the other task, dv_main, performs the verification function.

3.5.2 User Interface Design

The data verification utility is invoked from the HRS Main Menu Controller.

3.5.2.1 Definitions and Conventions

The conventions used in this section are defined in section 1.5 of this document.

3.5.2.2 DVT Main Window

When the data verification utility is invoked, its main window appears as illustrated in Figure 3-35.

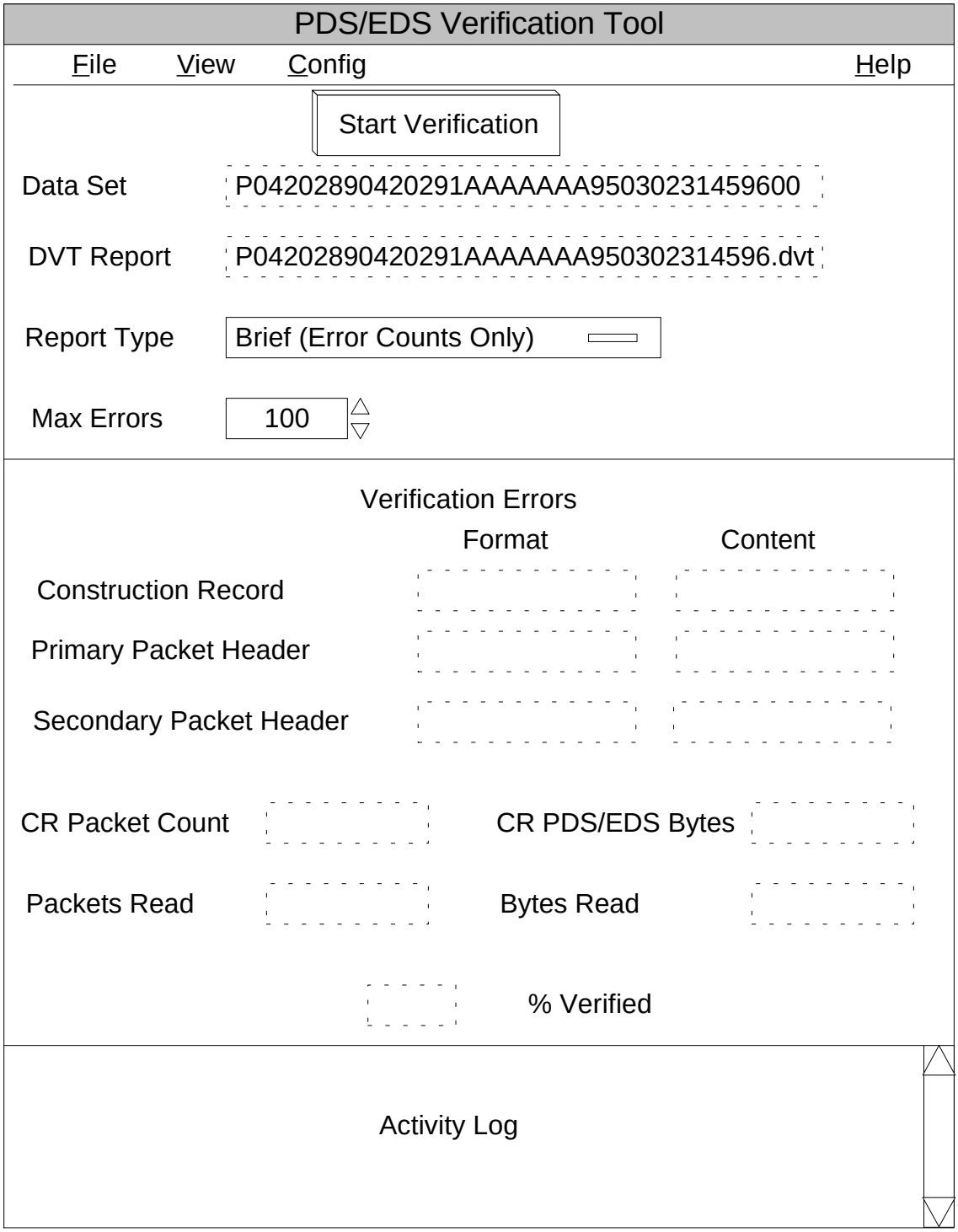

Figure 3-35. Main Window for Data Verification Tool

From the File Menu, the user selects the PDS or EDS to be verified. The Data Set and DVT Report fields are updated to display the file names of the PDS/EDS Construction Record and the tool's report file. The Report Type field has four choices on its option menu; Brief (Error Counts Only), Itemized Error Listing, Formatted Packet Headers, and Full (Error and Packet Headers). Optionally, the user may edit the "max errors" field and change any other default parameters for the tool. The "max errors" count allows the user to set an error limit that will stop the verification. The **Start Verification** button is used to start the processing. Once activated, the **Start Verification** button label toggles to **Stop Verification**, and the user may no longer make changes to the top portion of the display.

While the DVT is verifying the data set, it provides status to the operator. This includes current error counts by Construction Record, Primary and Secondary Packet Header categories. Format errors are violations of the field format, such as non-ASCII data in an ASCII field. Content errors are field values outside the allowed range of values and inconsistencies between CR and packet information . In addition to the error counts, the number of packets read and an estimate of the percent of data verified is given.

The operator may choose to stop the verification early. When the operator clicks on the **Stop Verification** button, he is prompted for confirmation. If the operator confirms the termination, he is then asked if the associated report should be completed.

Verification may also terminate early due to excessive errors being found. The operator may set a maximum error limit. When the error threshold is reached, the operator is notified via the activity log.

3.5.2.3 DVT Main Window Menu Bar

The Menu Bar for the DVT Main Window is shown in Figure 3-36.

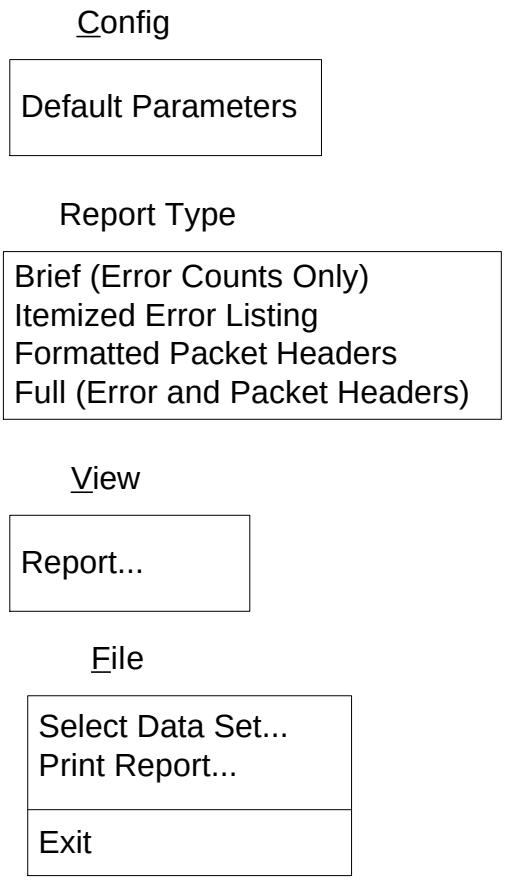

Figure 3-36. DVT Main Window Menu Bar

3.5.2.3.1 File

The File menu contains the following options:

Select Data Set... causes a file selection box to appear. Selecting a data set file from the list displayed and selecting **OK** causes the name of the first data set file to be shown on the display along with the data set's size. Since the report to be created is named after the data set, its file name will also be filled in. The report file name starts with the same first 34 characters of the data set name and uses ".dvt" as its file extension.

Print Report... causes a file selection box to appear. Selecting a report file from the list displayed and selecting **OK** causes the file to be sent to the system default printer.

Exit starts the shutdown sequence for the utility. If a verification is in progress, a confirmation box pops up with the message, "There is a verification in progress. Shutting down will cause the current report data to be lost. Continue with shutdown?" Selecting **Yes** in the confirmation box stops the verification and closes all files. Selecting **No** returns control to the main window. **No** is the default choice.

3.5.2.3.2 View

Selecting *Report...* causes a file selection box to appear. Selecting a report file from the list displayed and selecting **OK** causes the file to be sent to a scrollable window.

3.5.2.3.3 Config

Selecting *Default Parameters* causes a window to appear with all of the default parameters displayed. The parameters may be edited.

3.5.2.4 Default Parameters Window

The default parameters window (not shown), appears when the *Default Parameters* option is selected from the Config menu. These fields define expected values for fields such as the EDOS Software Version Number, Construction Record Type, Spacecraft Identification (SCID), Primary Packet Header Fields; Version Number, Type, and Secondary Header Flag. These values may be changed by the operator if necessary and saved.

3.5.3 Processing

The DVT performs three types of verifications on the selected data set. As the data set is read, fields are checked for possible format errors. Fields defined as unsigned integer cannot have format errors while fields defined as ASCII could contain non-ASCII values. If the ICD has further narrowed the possible values for a field, a valid range check is done. Lastly, verification is done of the information in the Construction Record to the packet headers. Inconsistencies between CR and packet information are counted as content errors.

First the Construction Record (CR) fields are checked for format and content errors. Certain field values are extracted for display, such as the size of the PDS/EDS in octets and the total PDS/EDS packet count. Several fields that point to data in the rest of the data set are saved for cross-checking with the packet information.

Next the primary and secondary packet headers are checked for format and content errors. As the packets are read, information from the CR can be verified. When all of the packet headers have been checked, the DVT report is produced.

Processing may be interrupted before the entire data set is read in several ways. The operator may control operation of the tool by setting a maximum error limit or may use the **Stop Verification** button on the main window. The DVT may terminate if there are problems in accessing the data set files.

At a minimum, the DVT report contains the name of the data set that was verified, the default parameter field values, the number of packets checked, an accounting of the types of errors found and the type of termination. If itemized errors were requested, the minimum report is followed by a listing of the errors identified by data set file number, offset into file, field name, field value and type of error. If formatted headers were requested, the report will contain the name of the formatted packet header file information. If the full report was selected, the formatted packet header lines and itemized error lines will be intermixed in the order they were processed.

3.5.4 Prologs/Program Design Language/ Structure Charts for Module Drivers

The structure charts for the data verification tool are shown in Figures 3-37 through 3-39.

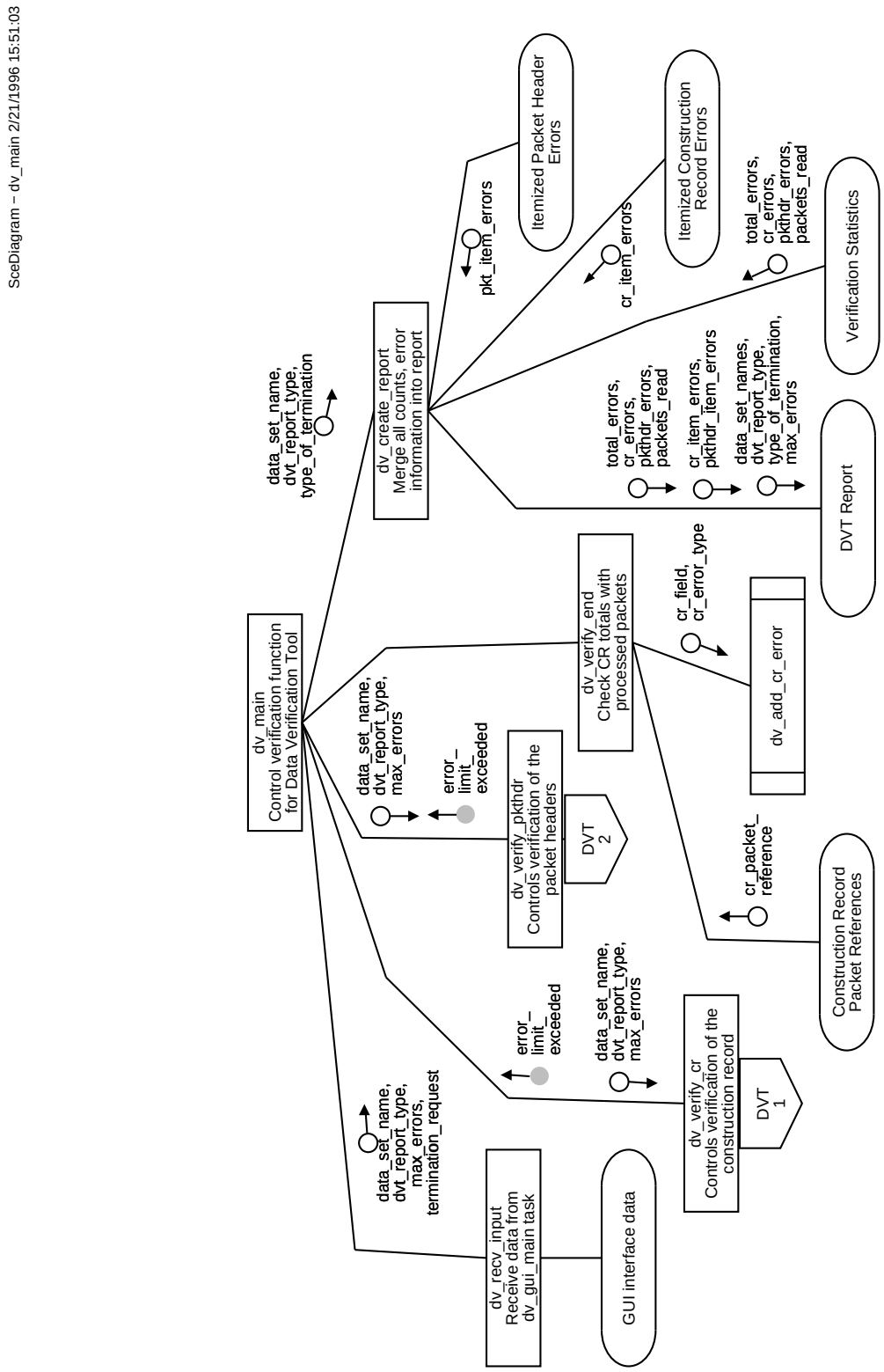

Figure 3-37. Structure Chart for Data Verification Tool (dv_main_task)

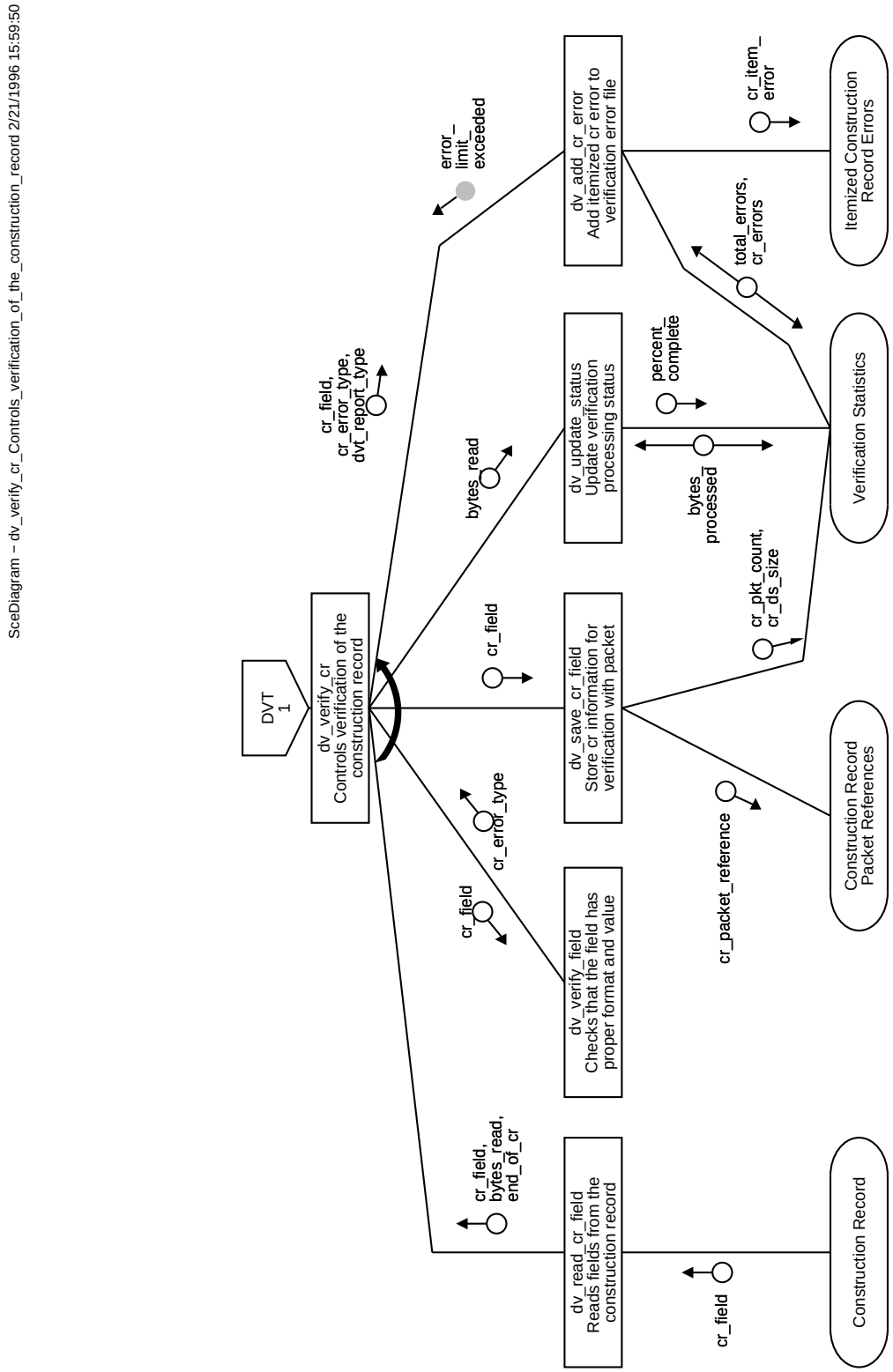

Figure 3-38. Structure Chart for Data Verification Tool (dv_verify_cr)

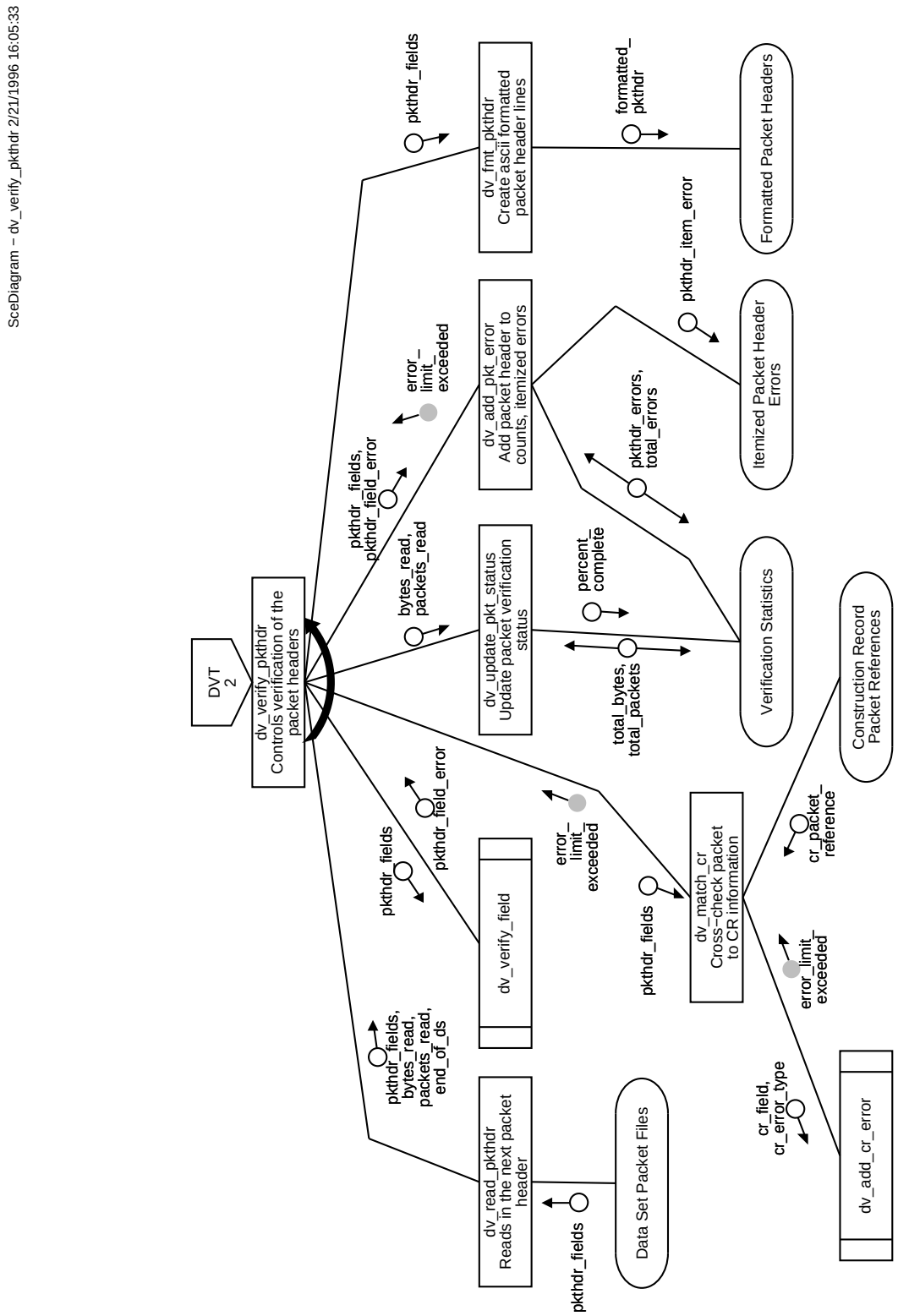

Figure 3-39. Structure Chart for Data Verification Tool (dv_verify_pkthdr)

3.5.4.1 dv_main

Purpose:

The dv_main procedure is the main routine for the data verification task.

The dv_main procedure controls the data set verification function. This task is activated by the dv_gui_main task when the operator pushes the **Start Verification** button. This task terminates when the operator pushes the **Stop Verification** button, the maximum number of errors has been found or the tool completes the verification of the data set.

External References:

dv_recv_input	Receives input from the dv_gui_main task
dv_verify_cr	Controls the verification of the construction record
dv_verify_pkthdr	Controls the verification of the packet headers
dv_verify_end	Verifies that last packet information from the construction record matches the last packet of the data set
dv_create_report	Merges all of the verification statistics with the detailed discrepancy information to create the report

Input:

Data Set to be verified

Maximum error limit

Report Type one of the following types of reports

- Brief (Error Counts Only)
- Itemized Error Listing
- Formatted Packet Headers
- Full (Error and Packet Headers)

Termination request from operator

Output:

DVT Report (named after the data set and has a file extension of ".dvt")

Requirements Met:

V1-2.1.5.1.1-1

PDL:

Call dv_recv_input to obtain data set name, maximum error limit and type of DVT report type from user interface task
Call dv_verify_cr to perform verification of the construction record
Call dv_verify_pkthdr to perform verification of the packet headers
Call dv_verify_end to verify CR total packet counts, data set size with processed counts
Call dv_create_report to merge final statistics, type of termination, and detailed error information into the DVT report file

3.5.4.2 dv_verify_cr

Purpose:

The dv_verify_cr procedure controls the verification of the construction record.

External References:

dv_read_cr_field	Reads fields from the construction record
dv_update_status	Updates verification processing status
dv_add_cr_error	Adds itemized construction record error record to the file
dv_save_cr_field	Stores construction record data for later verification with packet

Input:

Data set name
Maximum error limit
Report type
Termination request from operator
current total error count

Output:

construction record error counts
current total error count
detailed errors file

Requirements Met:

V1-2.1.5.1.2-1; V1-2.1.5.1.2-2; V1-2.1.5.1.2-3

PDL:

```
Until end of construction record is read or maximum error limit is reached or operator termination
        is requested
        Call dv_readnext_cr_field to read next field portion of construction record
        If the field has an invalid format
                Call dv_add_cr_error to update counts and create itemized error listing
        Else if the field value is out of range of ICD allowed values or equal to dvt default
                parameter value
                Call dv_add_cr_error to update counts and create itemized error listing
        Else if the field references a packet
                Call dv_save_cr_field to store information for later verification with referenced
                        packet header
        End if
        Calculate address of next field
        Call dv_update_cr_status to update bytes processed and percentage complete
End until
```

3.5.4.3 dv_verify_pkthdr

Purpose:

The dv_verify_pkthdr procedure controls the verification of packet header fields.

External References:

dv_read_pkthdr	Reads next packet header, opening next file in data set as needed
dv_add_pkt_error	Creates itemized error entry for report
dv_fmt_pkthdr	Creates ASCII formatted packet header lines for report
dv_match_cr	Compares packet information to construction record

Input:

Data set name

total error counts

termination request from operator

Output:

total error counts

bytes processed

appended itemized error file

error limit exceeded flag

Requirements Met:

V1-2.1.5.1.2-1; V1-2.1.5.1.2-2; V1-2.1.5.1.3-1

PDL:

```
Until end of data set is read or maximum error limit is reached or operator termination is
        requested
        Call dv_readnext_pkthdr to read next packet header
        If ASCII-formatted packet headers were requested
                Call dv_fmt_pkthdr to create ASCII-formatted header data
        Until all packet header fields are verified or maximum error limit is reached or operator
                termination is requested
                If the field has an invalid format
```

```
                    Call dv_add_pkt_error to update counts and create itemized error listing
            Else if the field value is out of range of allowed values
                    Call dv_add_pkt_error to update counts and create itemized error listing
            End if
        End until
        Call dv_update_pkt_status to update bytes processed and percentage complete
        Call dv_match_cr to verify packet referenced by the construction record
End until
```

3.5.4.4 dv_create_report

Purpose:

The dv_create_report procedure merges information from the processing statistics, error counts and detailed errors file into the final DVT report.

External References:

None

Input:

DVT report type

type of termination

total error count

bytes processed

packets processed

construction record error counts

packet header error counts

detailed errors file

formatted packet headers file

Output:

DVT Report File

Requirements Met:

V1-2.1.5.1.6-1; V1-2.1.5.1.6-2; V1-2.1.5.1.6-3; V1-2.1.5.1.6-4

PDL:

Create the DVT report file based on name of data set A and ".dvt" file extension

Write the maximum error limit and type of report requested to the DVT report file

Write the type of termination to the DVT report file

Write the full directory and file names of both data sets to the DVT report file

Write the file name of the ASCII formatted packet headers, if requested

Write number of files in data set, bytes and packets read, total error count to the DVT report file

Write the construction record error counts to the DVT report file

Write the packet header error counts to the DVT report file

If Itemized Error Listing was requested

    Append the itemized error file to the DVT report file

    Delete the itemized error file

End if

Close the DVT report file

3.6 Data Comparison Tool

3.6.1 Description

The Data Comparison Tool is present only on the High-Rate System. This tool takes two PDSs or EDSs as input and compares information in the construction records and in the packet headers. Discrepancies between the data sets are counted and optionally detailed in the resulting report.

The DCT is made up of two software processes. One task, dc_gui_main, controls the user interface and the other task, dc_main, performs the comparison function.

3.6.2 User Interface Design

The data comparison utility is invoked from the HRS Main Menu Controller.

3.6.2.1 Definitions and Conventions

The conventions used in this section are defined in section 1.5 of this document.

3.6.2.2 DCT Main Window

When the data comparison utility is invoked, its main window appears as illustrated in Figure 3-40.

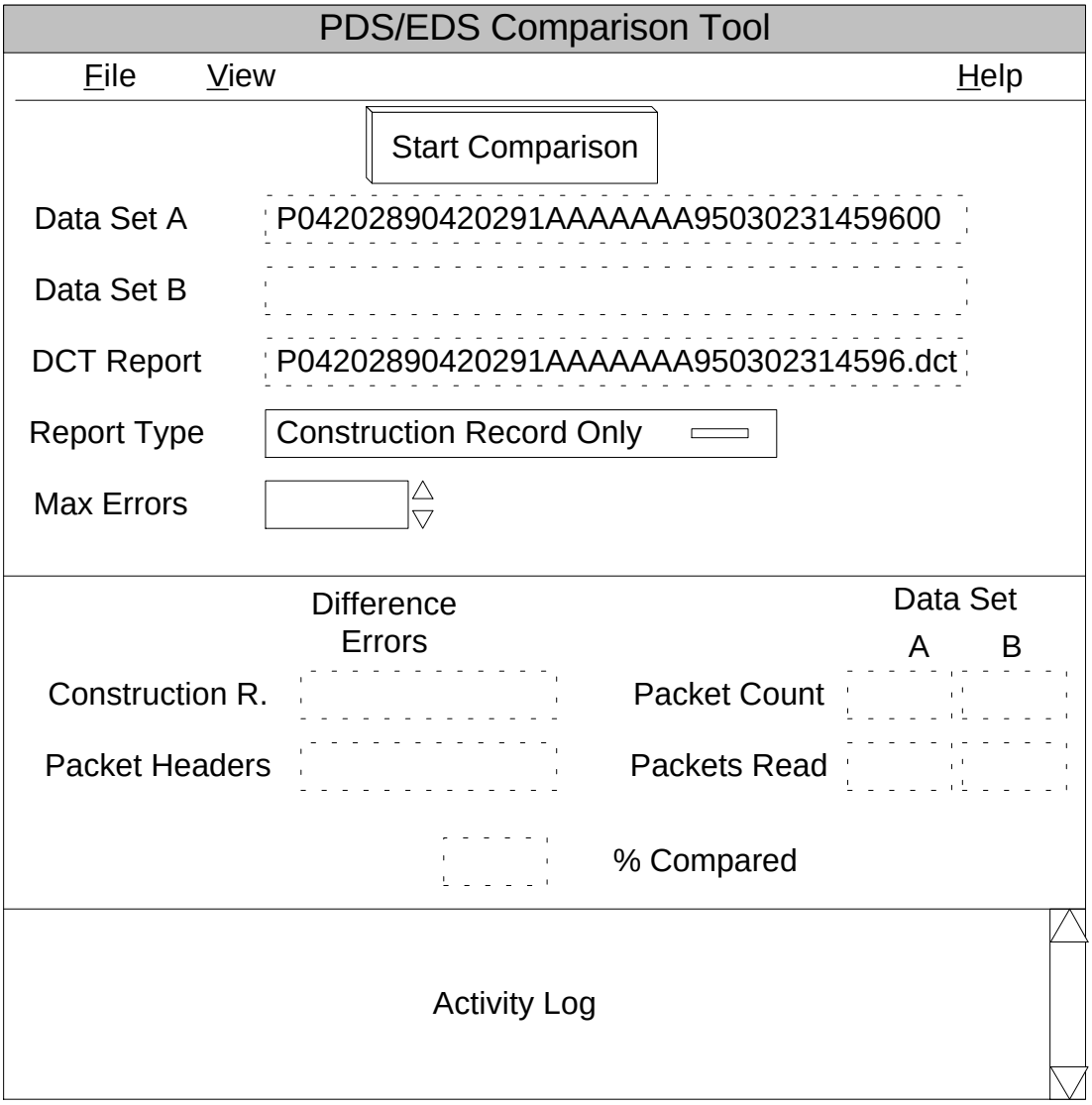

Figure 3-40. Main Window for Data Comparison Tool

From the File Menu, the user selects two PDSs or EDSs to be compared. The Data Set and DCT Report fields are updated to display the file names of the PDS/EDS Construction Records and the tool's report file. The Report Type field has three choices on its option menu; Brief (Difference Counts Only), Construction Record Only, and Full (CR and Packet Headers). The "max errors" count allows the user to set an error limit that will stop the comparison. The **Start Comparison** button is used to start the processing. Once activated, the **Start Comparison** button label toggles to **Stop Comparison**, and the user may no longer make changes to the top portion of the display.

While the DCT is comparing the data sets, it provides status to the operator. This includes current difference counts by Construction Record, Primary and Secondary Packet Header categories. In addition to the difference counts, the number of packets read and an estimate of the percent of data compared is given.

The operator may choose to stop the comparison at any time. When the operator clicks on the **Stop Comparison** button, he is prompted for a confirmation. If the operator confirms the termination, he is then asked if the associated report should be completed.

Comparison may also terminate early due to excessive differences being found. The operator may set a maximum error limit. When the error threshold is reached, the operator is notified via the activity log.

3.6.2.3 DCT Main Window Menu Bar

The Menu Bar for the DCT Main Window is shown in Figure 3-41.

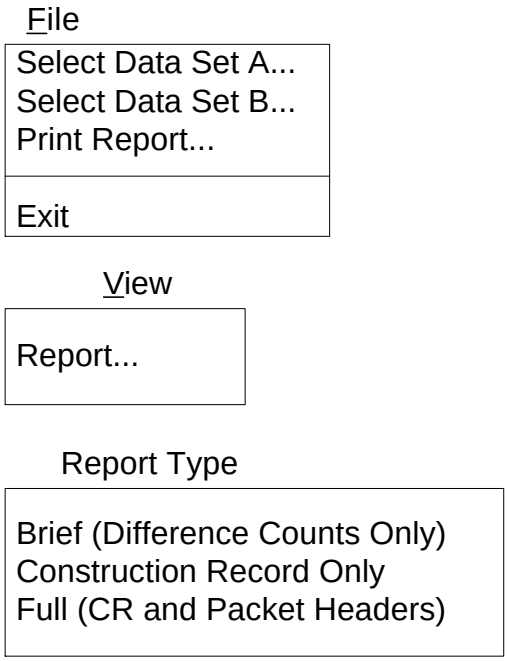

Figure 3-41. DCT Main Window Menu Bar

3.6.2.3.1 File

The File menu contains the following options:

Select Data Set A... causes a file selection box to appear. Selecting a data set file from the list displayed and selecting **OK** causes the name of the first data set file to be shown on the display along with the number of the data set's files. Since the report to be created is named after data set A, its file name will also be filled in. The report file name starts with the same first 34 characters of data set A name and uses ".dct" as its file extension.

Select Data Set B... causes a file selection box to appear. Selecting a data set file from the list displayed and selecting **OK** causes the name of the first data set file to be shown on the display along with the number of the data set's files.

Print Report... causes a file selection box to appear. Selecting a report file from the list displayed and selecting **OK** causes the file to be sent to the system default printer.

Exit starts the shutdown sequence for the utility. If a Comparison is in progress, a confirmation box pops up with the message, "There is a Comparison in progress. Shutting down will cause the current report data to be lost. Continue with shutdown?" Selecting **Yes** in the confirmation box stops the comparison and closes all files. Selecting **No** returns control to the main window. **No** is the default choice.

3.6.2.3.2 View

Selecting *Report...* causes a file selection box to appear. Selecting a report file from the list displayed and selecting **OK** causes the file to be sent to a scrollable window.

3.6.3 Processing

The Data Comparison Tool reads portions of the selected data sets in tandem and does a field by field comparison of the Construction Records and the primary and secondary packet headers. If requested, a detailed difference report is generated.

At a minimum, the DCT report contains the name of the data sets that were compared, the number of packets read, an accounting of the differences found and the type of termination. If the comparison was limited to the Construction Records, the minimum report is followed by a listing of the CR differences identified by data set file numbers, offsets into file, field names, and field values. If the full report was selected, all differences in the CRs and packet headers are provided.

3.6.4 Prologs/Program Design Language/Structure Charts for Module Drivers

The structure charts for the data comparison utility are shown in Figures 3-42 through 3-44.

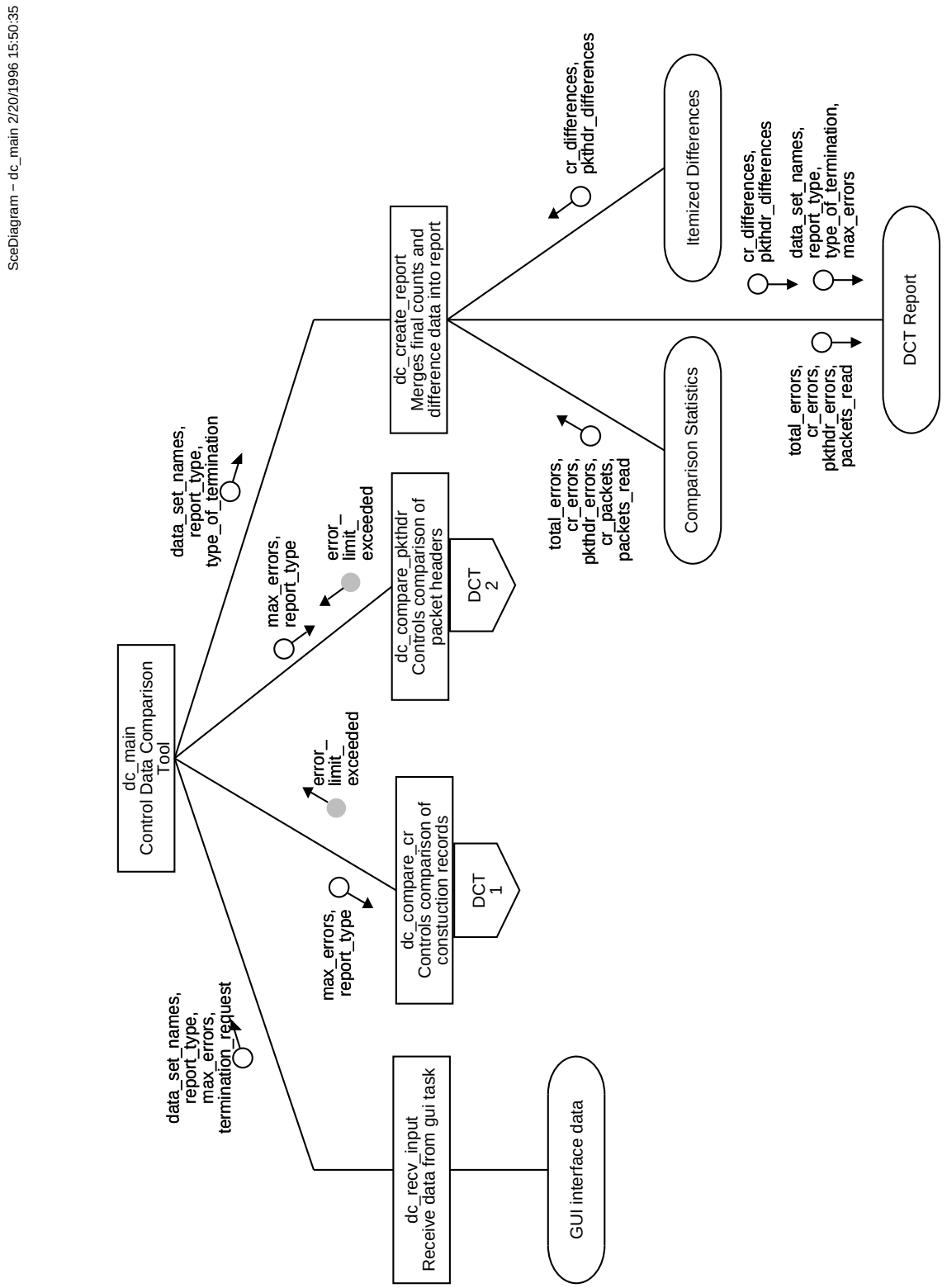

Figure 3-42. Structure Chart for Data Comparison Tool (dc_main_task)

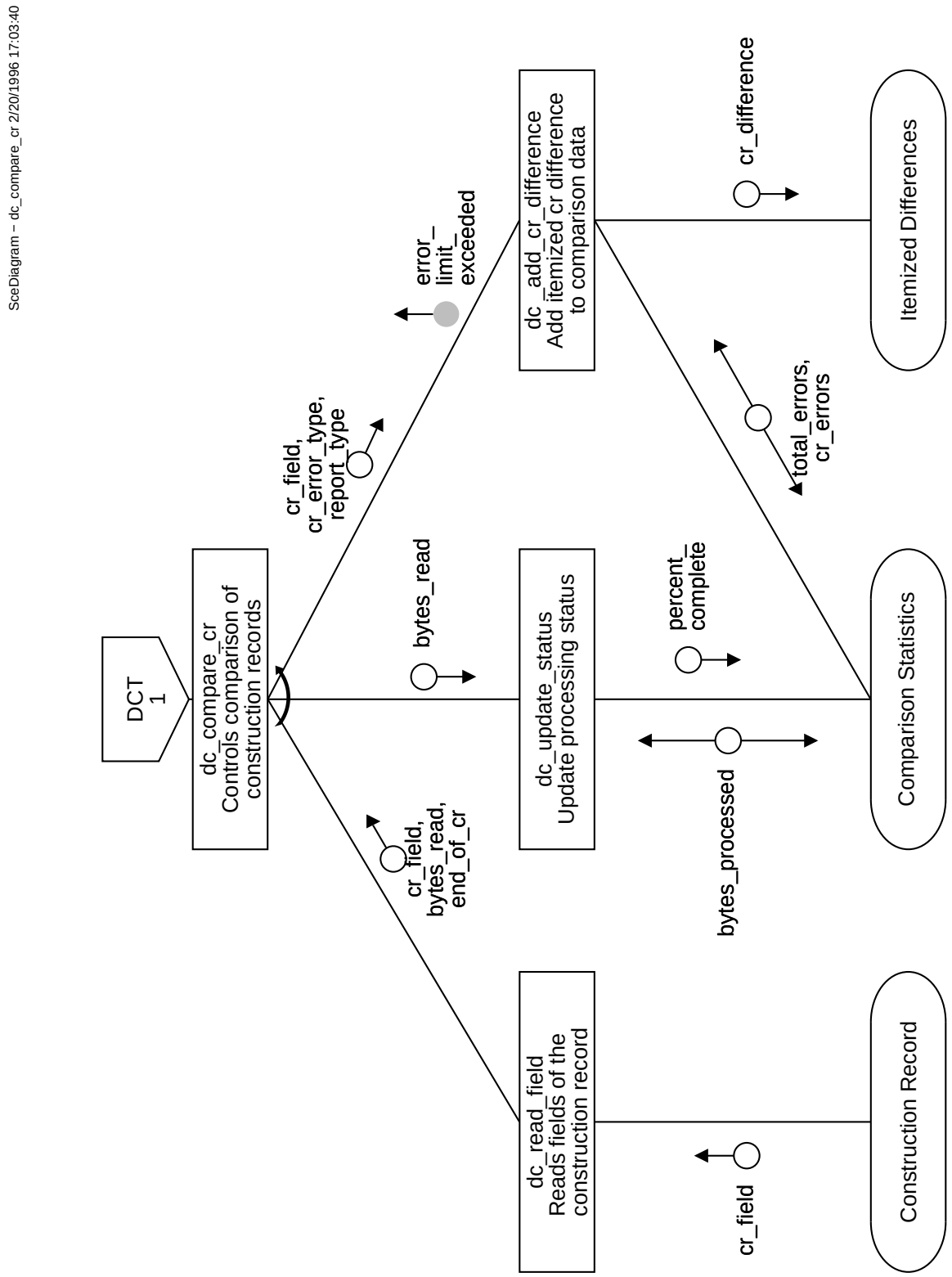

Figure 3-43. Structure Chart for Data Comparison Tool (dc_compare_cr)

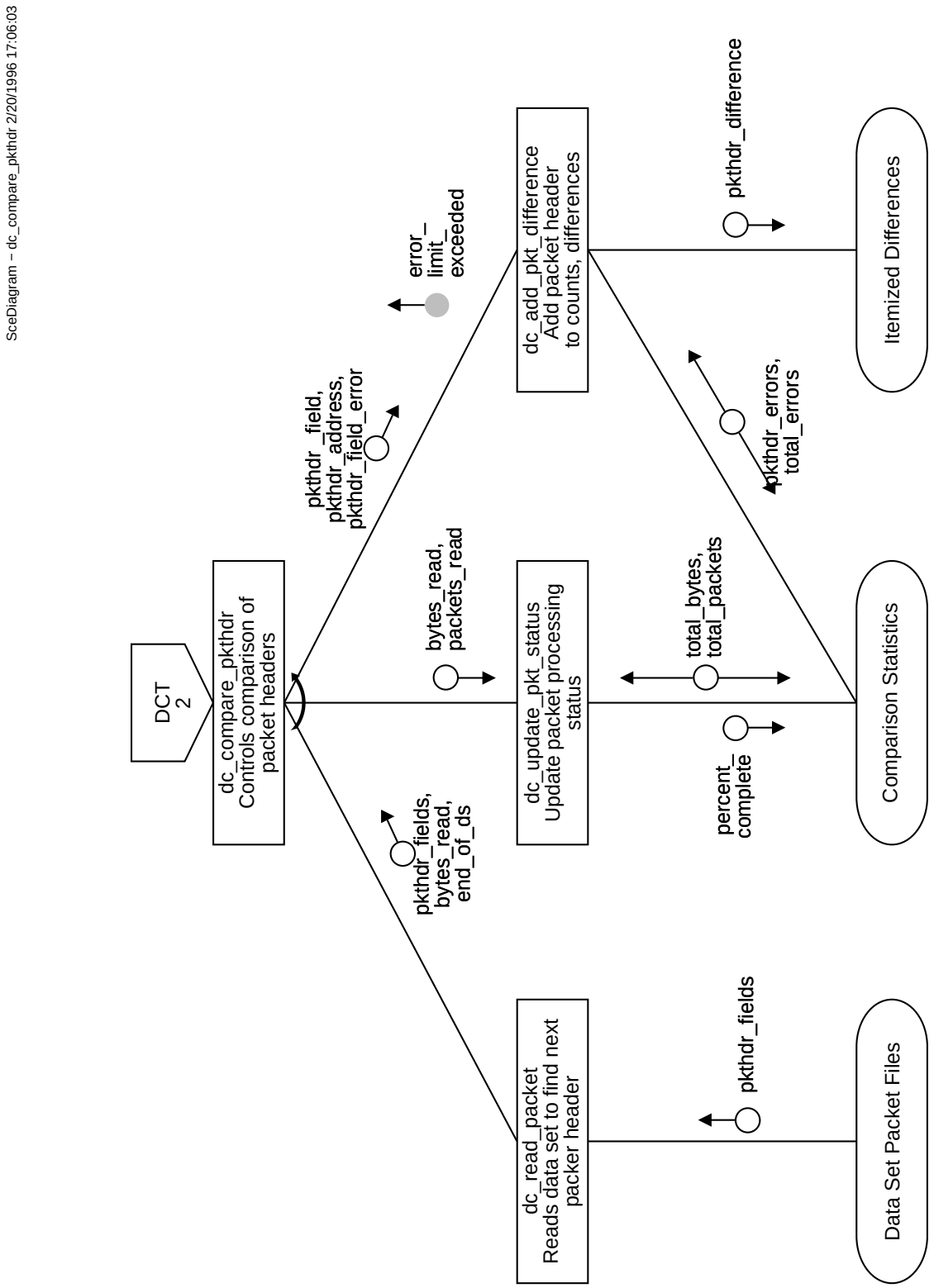

Figure 3-44. Structure Chart for Data Comparison Tool (dc_compare_pkthdr)

3.6.4.1 dc_main

Purpose:

The dc_main procedure controls the data set comparison function. This task is activated by the dc_gui_main task when the operator pushes the **Start Comparison** button. This task terminates when the operator pushes the **Stop Comparison** button, the maximum number of differences has been found or the tool completes the comparison of the data sets.

External References:

dc_compare_cr	Controls the comparison of the construction records
dc_compare_pkthdr	Controls the comparison of the packet headers
dc_create_report	Merges the final counts and differences into the DCT report
dc_recv_input	Receives input from the mc_gui_main task

Input:

Data Set A filename

Data Set B filename

Maximum error limit

Report Type - one of the following types of reports

- Brief (Difference Counts Only)
- Detailed Comparison of Construction Records Only
- Full (Detailed Comparison of Construction Records and Packet Headers)

Termination request from operator

Output:

DCT Report (named after the first data set and has a file extension of ".dct")

Requirements Met:

V1-2.1.5.2.1-1

PDL:

Call dc_recv_input to obtain data set names, maximum error limit and type of report from user interface task

Call dc_compare_cr to compare the fields of the construction records

Until maximum error limit is reached or operator termination is requested or the end of a data set is reached

Call dc_compare_pkthdr to compare primary and secondary packet headers

End until

Call dc_create_report to merge DCT configuration information, final difference statistics, type of termination and detailed difference information into the DCT report file

3.6.4.2 dc_compare_cr

Purpose:

The dc_compare_cr procedure compares fields from the construction records.

External References:

dc_read_field	Reads fields of the construction records
dc_update_status	Updates the processing status
dc_add_cr_difference	Adds an itemized construction record difference

Input:

Data Set A filename

Data Set B filename

Maximum error limit

Report Type - one of the following types of reports

- Brief (Difference Counts Only)
- Detailed Comparison of Construction Records Only
- Full (Detailed Comparison of Construction Records and Packet Headers)

Termination request from operator

current total error count

Output:

construction record error counts
current total error count
detailed differences file

Requirements Met:

V1-2.1.5.2.2-1; V1-2.1.5.2.2-2

PDL:

```
Until end of both construction records are is read or maximum error limit is reached or operator
        termination is requested

        Call dc_read_fieldrecord to read next portion of construction record from data set A

        Call dc_read_fieldrecord to read next portion of construction record from data set B

        Until all construction fields have been checked or maximum error limit is reached or
                operator termination is requested

                Extract next field from both construction records

                If the fields do not match

                        Call dc_add_cr_difference to add current field to error counts and detailed
                                difference file

                        End If

                End If

                Call dc_update_status to update bytes processed

        End until

End until
```

3.6.4.3 dc_compare_pkthdr

Purpose:

The dc_compare_pkthdr procedure compares fields from the primary and secondary packet headers.

External References:

dc_read_packet	Reads data set files in order to find the next packet header
dc_update_pkt_status	Updates the total counts for bytes and packets in each data set and calculates the percentage completed
dc_add_pkt_difference	Adds a packet error to the counts and generates the packet header difference listing

Input:

Data Set A filename

Data Set B filename

Maximum error limit

DCT Report Type - one of the following types of reports

- Brief (Difference Counts Only)
- Detailed Comparison of Construction Records Only
- Full (Detailed Comparison of Construction Records and Packet Headers)

Termination request from operator

current total error count

Output:

packet header error counts

current total error count

appended detailed differences file

bytes processed

packets processed

Requirements Met:

V1-2.1.5.2.3-1; V1-2.1.5.2.3-2; V1-2.1.5.2.4-1

PDL:

```
Until an end of file for both data sets is reached or maximum error limit is reached or operator
        termination is requested

        Call dc_read_packet to read next packet header from both data sets

        Until all primary and secondary packet header fields have been checked or maximum error
                limit is reached or operator termination is requested

                Extract next packet header field from both data set read buffers

                If the fields do not match

                        Call dc_add_pkt_difference to add current field to error counts and
                                detailed difference file

                End If

                Call dc_update_pkt status to update packet bytes processed, packets processed,
                        percent complete

        End until

End Until

If one end of file was reached and a packet header was found for the other data set

        Until end of file is reached for the remaining data set

                Call dc_read_packet to read next packet header from the data set

                Call dc_add_pkt_difference to add each field to error counts and detailed
                        difference file

                Call dc_update_pkt_status to update bytes processed, packets processed, percent
                        complete

        End Until

End If
```

3.6.4.4 dc_create_report

Purpose:

The dc_create_report procedure merges information from the processing statistics, error counts and detailed differences file into the final DCT report.

External References:

Detailed Differences File

DCT Report File

Input:

Data Set A filename

Data Set B filename

Maximum error limit

DCT report type

type of termination

Output:

DCT Report File

Requirements Met:

V1-2.1.5.2.4-2; V1-2.1.5.2.5-1; V1-2.1.5.2.5-2; V1-2.1.5.2.5-3

PDL:

Create the DCT report file based on name of data set A and ".dct" file extension

Write the maximum error limit and type of report requested to the DCT report

Write the type of termination to the DCT report file

Write the full directory and file names of both data sets to the DCT report file

Write number of files in each data set, bytes and packets read, total error count to the DCT report file

Write the construction record difference counts to the DCT report file

Write the packet header difference counts to the DCT report file

If Detailed Comparisons were requested

    Append the detailed differences file to the DCT report file

    Delete the detailed differences file

End if

Close the DCT report file

Section 4. Size Estimates

Table 4-1. Support Software Code Estimates

Component	DSI	Reuse %	ETS Platform
Menu Controller	2500	0	Development for MPS, LRS and HRS
OMDSIM	5000	0	Development for MPS, LRS and HRS
Rate Buffering Utility	2200	0	Development for LRS
Command Archive	2000	0	Development for LRS
Data Verification Tool	4000	0	Development for HRS
Data Comparison Tool	3000	0	Development for HRS
Total	18700		= 18700 wDSI

Appendix A. Support Software Requirements to Design Traceability

This matrix provides a "forward" reference from the Support Software level 4 detailed requirements to design element names as identified in this detailed design document. The matrix is in level 4 requirement number order in the following format:

- Level 4 Req Spec—Support Software level 4 requirement specification number contained in the ETS Requirements Specification Volume 1 document, dated xx/xx/95 (Reference 4)

  NOTE: A requirement tagged as [MOD] represents a requirement which has been modified from the level 4 requirement contained in the requirements specification document. A requirement tagged as [DEL] represents a requirement which has been deleted from the requirements specification document. Until the requirements specification document (Reference 4) is updated, the level 4 requirements given in the following table supersede those given in the requirements specification document.

- Requirement description—The textual representation of this requirement as written in the ETS Requirements Specification Volume 1 document. (See note above concerning requirements tagged as [MOD] or [DEL].)

- Design element — The software CI, process, subprocess or subroutine that satisfies the requirement.

- Build — The system/build which satisfies this requirement.

NOTE: Each volume in this five-volume set contains its own level 4 traceability matrix which provides a design cross reference for the design elements contained in that specific volume.

Level 4 Req Spec	Requirement Description	Design Element	Build
V1-1.1.1-1	The ETS Menu Controller shall provide a GUI to all ETS integrated software products.	Menu Controller CI	LRS/1 MPS/1 HRS/1
V1-1.1.1-2	The ETS Menu Controller shall comply with the Open Software Foundation (OSF) Motif Style Guide, revision 1.2 (Reference 8).	Menu Controller CI	LRS/1 MPS/1 HRS/1
V1-1.1.1-3	The ETS Menu Controller shall provide the capability to activate, monitor, and terminate processes associated with ETS operations.	Menu Controller CI	LRS/1 MPS/1 HRS/1
V1-1.1.1-4	The ETS Menu Controller shall provide the capability to activate and terminate the SCTGEN.	mc_display_menu mc_start_appl mc_stop_appl	LRS/1 MPS/1 HRS/1

Level 4 Req Spec	Requirement Description	Design Element	Build
V1-1.1.1-5	The ETS Menu Controller shall provide the capability to activate and terminate the TPCE for the VME High-Rate Subsystem (VHS).	mc_display_menu mc_start_appl mc_stop_appl	LRS/1 MPS/1 HRS/1
V1-1.1.1-6	The ETS Menu Controller shall provide the capability to activate and terminate the TPCE for the VME Low-Rate Subsystem (VLS).	mc_display_menu mc_start_appl mc_stop_appl	LRS/1 MPS/1 HRS/1
V1-1.1.1-7	The ETS Menu Controller shall provide the capability to activate and terminate the OMD Simulator.	mc_display_menu mc_start_appl mc_stop_appl	LRS/1 MPS/1 HRS/1
V1-1.1.1-8	The ETS Menu Controller shall provide the capability to activate and terminate the CDS Utilities.	mc_display_menu mc_start_appl mc_stop_appl	LRS/1 MPS/1 HRS/1
V1-1.1.1-9	The ETS Menu Controller shall visibly acknowledge a request from a local user within 2 seconds of its entry.	XtAppMainLoop	LRS/1 MPS/1 HRS/1
V1-1.1.1-10	The ETS Menu Controller shall start the execution of a local user request within 5 seconds of its entry.	XtAppMainLoop	LRS/1 MPS/1 HRS/1
V1-1.1.1-11	Deleted	n/a	n/a
V1-1.1.3.1-1	The ETS OMDSIM shall read a store of OMD and make pregenerated OMD available for transmission.	om_get_default_OMD om_create_gui, om_init_gui om_scenario_manager om_scheduler, om_transmission_task op_input_communication om_get_default_OMD	LRS/1 MPS/1 HRS/1
V1-1.1.3.1-2 [MOD]	The ETS OMDSIM shall display all modifiable fields of the OMD to the user and allow the operator to modify these fields.	om_init_giu, om_create_gui,	LRS/2 MPS/2 HRS/2
V1-1.1.3.1-3	The ETS OMDSIM shall allow the user to store OMD to a file (or other storage device).	om_init_gui, om_get_default_OMD op_input_communication	LRS/1 MPS/1 HRS/1
V1-1.1.3.2-1	The ETS OMDSIM shall provide the capability to receive OMD through EBnet.	om_reception_task	LRS/1 MPS/1 HRS/1
V1-1.1.3.3-1	The ETS OMDSIM shall respond to OMD received from other EOSDIS test elements with an acknowledgment or disposition OMD, as appropriate.	om_reception_task	LRS/2 MPS/2 HRS/2

Level 4 Req Spec	Requirement Description	Design Element	Build
V1-1.1.3.4-1	The ETS OMDSIM shall display a record of OMD transmission events, which include: a. Time of transmission b. Type of OMD transmitted c. Destination of OMD d. Sequence number	om_transmisison_task, om_logger, XtAppMainLoop	LRS/2 MPS/2 HRS/2
V1-1.1.3.4-2	The ETS OMDSIM shall display the OMD receipt event, which consists of: a. Time of receipt b. Type of OMD received c. Source of OMD d. Sequence number	om_reception_task, om_logger, XtAppMainLoop	LRS/2 MPS/2 HRS/2
V1-1.1.3.4-3	The ETS OMDSIM shall display accounting information. This information includes: a. Total number of OMD transmitted b. Total number of OMD received c. Number of acknowledgments and the number of acknowledgments expected but not received (nacks)	om_reception_task, om_transmission_task, om_logger, XtAppMainLoop	LRS/2 MPS/2 HRS/2
V1-1.1.3.5-1	The ETS OMDSIM shall transmit OMD to the EOSDIS test elements through EBnet.	om_send_to_external, om_transmission_task	LRS/1 MPS/1 HRS/1
V1-1.1.3.6-1	The ETS OMDSIM shall provide the capability to log the OMD transmission event. ETS OMDSIM will log the time of the transmission, the OMD transmitted, and the transmitted OMD sequence number.	om_logger	LRS/1 MPS/1 HRS/1
V1-1.1.3.6-2	The ETS OMDSIM shall provide the capability to log the OMD received. ETS OMDSIM shall log the time of the receipt of the OMD, the OMD received, and the received OMD sequence number.	om_logger	LRS/1 MPS/1 HRS/1
V1-1.1.3.7-1	The ETS OMDSIM shall delog user selected portions of the OMD log file. All OMD fields can be selected by the user	om_delogger	LRS/2 MPS/2 HRS/2

Level 4 Req Spec	Requirement Description	Design Element	Build
V1-1.1.3.8-1	The ETS OMDSIM shall simulate CODAs using real time status data provided by the LRS hardware.	om_transmission_task	LRS/2
V1-1.1.3.8-2	The ETS OMDSIM shall allow the user to introduce errors into OMD headers and data fields prior to transmission. Error types include: a. erroneous values b. internal inconsistencies	om_transmission_task, om_calculate_element, om_scenario_manager	LRS/2 MPS/2 HRS/2
V1-1.1.3-1	The ETS OMDSIM shall provide the capability to receive OMD at rates up to 50 kilobits per second (Kbps).	om_reception_task	LRS/1 MPS/1 HRS/1
V1-1.1.3-2	The ETS OMDSIM shall provide the capability to generate and transmit OMD at rates up to 50 Kbps.	om_transmission_task, om_scheduler	LRS/2 MPS/2 HRS/2
V1-1.1.4.1-1 [MOD]	The ETS MRBD shall communicate with the VLS via Transmission Control Protocol (TCP)/Internet Protocol (IP) sockets.	rb_init_comm rb_recv_signal rb_receive_data	LRS/2
V1-1.1.4.1-2	The ETS MRBD shall write each message it receives over the socket to the file.	rb_write_data	LRS/2
V1-1.1.4.1-3	The ETS MRBD shall be capable of termination via operator directive.	rb_recv_signal	LRS/1
V1-1.1.4.1-4	The ETS MRBD shall terminate processing upon receiving a close socket indication from the VLS.	rb_recv_signal	LRS/2
V1-1.1.4.1-5 [MOD]	The ETS MRBD shall perform the following functions to terminate processing: a. close the socket b. close the file	rb_recv_signal	LRS/1
V1-1.1.5.1-1 [NEW]	The ETS Command Archive Utility shall communicate with the VLS via TCP/IP sockets.	command archive utility	LRS/2
V1-1.1.5.1-2 [NEW]	The ETS Command Archive Utility shall write each Command Data Block it receives to an archival file.	command archive utility	LRS/2
V1-2.1.5.1.1-1	The ETS DVT shall accept the following types of files to be verified: a. Expedited Data Set (EDS) b. Production Data Set (PDS)	dv_main task	HRS/2

Level 4 Req Spec	Requirement Description	Design Element	Build
V1-2.1.5.1.1-2 [MOD]	The ETS DVT shall open each and every file associated with the data set, reading all file records.	dv_read_field dv_read_pkthdr dv_verify_end	HRS/1
V1-2.1.5.1.2-1	Fields shall be verified by comparison to the expected value in the ETS DVT configuration file. These fields include: a. 'Software Version Number' b. 'Construction Record Type' c. Spacecraft ID	dv_verify_cr	HRS/1
V1-2.1.5.1.2-2	Fields shall be verified by comparison to a valid value/range as given in the DFCD. These fields include: a. the 'Location of first fill octet for packet'	dv_verify_cr dv_verify_pkthdr	HRS/2
V1-2.1.5.1.2-3	Fields shall be verified by ensuring consistency between a count and the list of items in the Construction Record following the count. These fields include: a. 'Number of APIDs in the data set' is consistent with the number of items in the list of APIDs. b. 'Number of TSS start/stop times' is consistent with the number of items in the list of start/stop times. c. 'Number of missing packet Source Sequence Count' is consistent with the number of items in the list of missing packets. d. 'Number of packets in list of packets containing EDOS generated fill data' is consistent with the list of fill data packet items.	dv_verify_cr	HRS/1

Level 4 Req Spec	Requirement Description	Design Element	Build
V1-2.1.5.1.3-1	Fields shall be verified by comparison to the expected value in the ETS DVT configuration file. These fields include: a. 'Version Number' of the packet header b. 'Type' of the packet header c. 'Secondary Header Flag' of the packet header d. 'Sequence Flag' of the packet header	dv_verify_pkthdr	HRS/2
V1-2.1.5.1.3-2 [DEL]	The ETS DVT shall count the 'Number of length errors' by verifying that a valid Path SDU appears where expected in the file. The DVT shall use the 'Packet Length' from the current Path SDU packet primary header to check for the expected Path SDU header to appear in the appropriate place in the file. If the ETS DVT is not able to find a valid Path SDU easily, then the tool shall terminate with an indication of the last valid Path SDU and its location in the file.	n/a	n/a
V1-2.1.5.1.4-1 [DEL]	The ETS DVT shall verify that the last record in the data set contains the appropriate end record identification.	n/a	n/a
V1-2.1.5.1.4-2 [DEL]	The ETS DVT shall verify that the last record in the data set contains the same data set identification as the Construction Record of the data set.	n/a	n/a
V1-2.1.5.1.5-1	The ETS DVT shall verify that the APID in the primary packet header for each Path service data unit is in the list of Construction Record APIDs and that the 'index to the first packet in the data set for APID' is correct.	dv_match_cr	HRS/2
V1-2.1.5.1.5-2	The ETS DVT shall verify that a packet is missing by checking the 'Packet Sequence Count' in the current Path SDU packet primary header against the previous one for that APID.	dv_match_cr, dv_update_pkt_status	HRS/2
V1-2.1.5.1.5-3	The ETS DVT shall verify that a missing packet is in the list of missing packets.	dv_match_cr	HRS/2

Level 4 Req Spec	Requirement Description	Design Element	Build
V1-2.1.5.1.5-4	The ETS DVT shall verify the 'Count of packets in the data set' by counting the number of packets in the data set.	dv_update_pkt_status, dv_match_cr	HRS/2
V1-2.1.5.1.5-5	The ETS DVT shall verify that the Construction Record 'Location of the first octet of generated fill data' points to an octet containing the fill pattern. The fill pattern is contained in the ETS DVT configuration file.	dv_match_cr	HRS/2
V1-2.1.5.1.5-6	The ETS DVT shall verify the 'Count of octets of EDOS generated fill data' by calculations using packet length and start of fill location.	dv_match_cr	HRS/2
V1-2.1.5.1.5-7	The ETS DVT shall verify that the CCSDS timecode in the secondary header of the first packet in the data set matches that in the Construction Record.	dv_match_cr	HRS/2
V1-2.1.5.1.5-8	The ETS DVT shall verify that the CCSDS timecode in the secondary header of the last packet in the data set matches that in the Construction Record.	dv_match_cr	HRS/2
V1-2.1.5.1.5-9 [MOD]	The ETS DVT shall verify the 'data set size (in bytes)' in the Construction Record by counting the number of bytes in the data set, exclusive of the Construction Record.	dv_update_pkt_status, dv_verify_end	HRS/1
V1-2.1.5.1.6-1	The ETS DVT shall write an ASCII-formatted report to disk.	dv_create_report	HRS/1
V1-2.1.5.1.6-2	The name of the report shall be created by appending ".dvt" to the input filename.	dv_create_report	HRS/1
V1-2.1.5.1.6-3 [MOD]	The report shall include: a. Construction Record formatted to show a subset of the fields b. Indication of the integrity of the structure of the Construction Record c. Comparison of Construction Record to actual Data Set	dv_create_report	HRS/2

Level 4 Req Spec	Requirement Description	Design Element	Build
V1-2.1.5.1.6-4	The ETS DVT shall also, as an option, write another file that contains an ASCII-formatted subset of the packet headers, which can be used for display and browse purposes.	dv_create_report	HRS/2
V1-2.1.5.2.1-1 [MOD]	The ETS DCT shall accept the following types of data sets to be compared: a. Expedited Data Set (EDS) b. Production Data Set (PDS	dc_main task	HRS/1
V1-2.1.5.2.1-2 [MOD]	The ETS DCT shall open each and every file associated with the data sets, reading all file records.	dc-read_record dc_read_packet	HRS/1

Level 4 Req Spec	Requirement Description	Design Element	Build
V1-2.1.5.2.2-1	The ETS DCT shall compare the following fields of EDS/PDS Construction Records: a. Software Version Number b. CR Type c. Number of APIDs in the data set d. List of APIDs in the data set e. VCDU-ID associated with APIDs f. Number of TSS start/stop times g. Number of missing packet Source Sequence Counts h. List of missing packets i. Count of packets from VCDUs with errors corrected by RS decoding j. Count of packets k. Data set size (in octets) l. Number of packets in list of packets containing EDOS generated fill data m. List of packets with fill data n. Count of octets of fill data o. Count of packets with discrepancies between packet header length and actual packet length p. CCSDS binary timecode in the secondary header of the first packet in DS q. CCSDS binary timecode in the secondary header of the last packet in DS	dc_compare_cr	HRS/1
V1-2.1.5.2.2-2	The ETS DCT shall write the values of the fields from both data sets if they do not match.	dc_compare_cr, dc_add_cr_difference, dc_create_report	HRS/1
V1-2.1.5.2.3-1	The ETS DCT shall compare the all the fields in the Packet primary and secondary headers.	dc_compare_pkthdr	HRS/2
V1-2.1.5.2.3-2	The ETS DCT shall write the values of the fields from both data sets if they do not match.	dc_compare_pkthdr, dc_add_pkt_difference, dc_create_report	HRS/1

Level 4 Req Spec	Requirement Description	Design Element	Build
V1-2.1.5.2.4-1 [DEL]	The ETS DCT shall check that the EDS/PDS End Record occurs at the same place in each data set.	n/a	n/a
V1-2.1.5.2.4-2 [DEL]	The ETS DCT shall write the location of the EDS/PDS End Record of both data sets, if they do not match.	n/a	n/a
V1-2.1.5.2.5-1	The ETS DCT shall write an ASCII-formatted report to disk.	dc_create_report	HRS/1
V1-2.1.5.2.5-2	The ETS DCT shall create the name of the report by appending ".dct" to the base data set name.	dc_create_report	HRS/1
V1-2.1.5.2.5-3 [MOD]	The ETS DCT report shall include information such as: a. names of the two files b. differences between the two Construction Records c. differences between the two sets of packet headers	dc_create_report	HRS/2
V1-8.1.1-1	ETS shall comply with the set of display guidelines specified in DSTL-92-007, Human-Computer Interface Guidelines, August 1992 (Reference 4).	mc_main, dc_gui_main, dv_gui_main, rb_main, ca_main tasks	
V1-8.1.2-1	ETS equipment shall operate in operational ambient temperature to range from 50 degrees to 80 degrees Fahrenheit and ambient humidity to range from 20 percent to 70 percent (noncondensing).		
V1-8.1.2-2	The ETS equipment shall require a physical space of no more than 10 feet by 20 feet.		
V1-8.1.2-3	The ETS equipment shall require approximately 10 kilowatts (±10 percent) for power.		
V1-8.1.3-1	ETS shall provide sensitivity level 1 protection as specified in the MO&DSD *Automated Information System (AIS) Security Policy* (Reference 5).		
V1-8.1.3-2	ETS shall comply with the *NASA Communications (Nascom) Access Protection Policy and Guidelines* (Reference 6).		

Appendix B. Data Dictionary

accounting_statistics	number of OMD received and transmitted.
Archive status	This information includes the currently open archive file names and sizes.
Command Archive Files	Archive files will be created for each TSS supported by the LRS. One file will contain all of the command blocks transferred to the workstation for archiving. The second file will only contain command blocks that failed validation. The naming conventions for these files will incorporate the session start time and type of file. cmd_archive_all_yydddhhmmss and cmd_archive_invalid_yydddhhmmss
GUI widgets	This is all the information needed in support of a graphical user interface, all the tables, alternate mouse pointer icons, graphical representations of all the windows, menus, dialog boxes, etc. This information will be generated with the BuilderXcessory tool.
icon	Graphical identification of a software component.
icon action	A selection from an icon menu. The options are *Restore, Move, Lower, and Close*
Icon descriptor table	This table defines for each icon the files that hold the icon image and activation scripts, the applicable ETS system, and whether or not multiple copies of the associated software may be run.
log_data_type	type indication of data to be logged (OMD_received, OMD_transmitted, OMD_event)
omd_elements	individual components of OMD data structures
OMD_element_rate_data	rates for one-up counter incrementation for all possible counter values for all OMD (also, one-up counters).
OMD_bitstream	transmitted OMD information
Rate Buffered Files	Rate buffered files will be created for each TSS for which rate buffering is enabled. The files have a specific naming convention as specified in the EDOS-EGS ICD (Reference 10).
rate buffering status	Status for the rate buffered data utility consists of the current file names, file sizes.
(OMD) type	indication of OMD type (CODA, TSS, EDS delivery record, PDS delivery record)
ready_for_next_record	return signal to indicate end of current portion of OMD scenario

start_transmission	signal to start OMD transmission scheduled by scenario file.
stop_transmission	signal to normally terminate OMD transmission scheduled by scenario file.
system name	Identifies which ETS system is being started up. The values are HRS, LRS and MPS.
tranmission_interval	The time interval (seconds) between successive transmissions of a single type of OMD in the schedule.
transmission_duration	The length of time for which a schedule for transmission of a single OMD type is maintained.
transmit_OMD	a bit signal requesting the transmission of the currently loaded OMD to the external EBnet interface.
working_OMD	The data structure containing a union of records that contain 1) the default and static values for OMD, two bits of selection information for each value that may be updated prior to transmission by a) LRS H/W, b) OMDSIM (the one-up counters).

Appendix C. Abbreviations and Acronyms

AM-1	morning equatorial crossing spacecraft series
AOS	Advanced Orbiting Systems
APID	application identifier
ATS	absolute time sequence
BER	bit error rate
bps	bits per second
BX	BuilderXcessory
C&DH	command and data handling
C&S	control and status
CADU	channel access data unit
CCB	Configuration Control Board
CCR	configuration change request
CCSDS	Consultative Committee for Space Data Systems
CDS	Control and Display Subsystem
CI	configuration item
CLCW	command link control word
CLTU	command link transmission unit
CODA	customer operations data accounting
COP	command operations procedure
COTS	commercial off-the-shelf
CR	construction record
CSC	Computer Sciences Corporation
CVCDU	coded virtual channel data unit
DAAC	Distributed Active Archive Center
DCT	Data Comparison Tool
DD	data dictionary
DDS	detailed design specification

DFCD	data format control document
DFD	data flow diagram
DS	data set
DSN	Deep Space Network
DSTL	Data Systems Technology Laboratory
DVT	Data Verification Tool
EBnet	EOSDIS Backbone network
EDOS	EOS Data and Operations System
EDS	expedited data set
EDU	EDOS data unit
EGS	EOS Ground System
EOC	EOS Operations Center
EOS	Earth Observing System
EOSDIS	Earth Observing System Data and Information System
ESDIS	Earth Science Data and Information System
ETS	EOSDIS Test System
EU	engineering unit
F&PR	functional and performance requirements
FARM	frame acceptance and reporting mechanism
FOT	Flight Operations Team
FTP	file transfer protocol
GMT	Greenwich mean time
GN	Ground Network
GSFC	Goddard Space Flight Center
GUI	graphical user interface
HR	High Rate
HRS	High-Rate System
HW	hardware
IDR	interface requirements document

IP	internet protocol
ISO	International Standards Organization
IV&V	independent verification and validation
kbps	kilobits (thousands of bits) per second
LR	Low Rate
LRS	Low-Rate System
Mbps	megabits (millions of bits) per second
MDM	multiplexer-demultiplexer
MET	mission elapsed time
MPS	Multimode Portable Simulator
MRBD	managed rate buffered data
MSB	Microelectronic Systems Branch
NASA	National Aeronautics and Space Administration
Nascom	NASA Communications
OMD	operations management data
OMDSIM	OMD Simulator
PDB	project database
PDL	program design language
PDS	production data set
PN	pseudorandom noise
Pspec	process specification
PSS	portable spacecraft simulator
RB	rate buffered
RF SOC	Radio Frequency Simulations Operations Center
RTS	relative time sequence
SCTGEN	Simulated CCSDS Telemetry Generator
SDR	system design review
SDU	service data unit
SN	Space Network

SOC	Simulations Operations Center
SRS	system requirements specification
SSR	solid state recorder
StP	Software through Pictures
SW	software
TBC	to be confirmed
TBD	to be determined
TBS	to be specified
TC	telecommand
TCP	transmission control protocol
TDRS	Tracking and Data Relay Satellite
TDRSS	Tracking and Data Relay Satellite System
TGT	TDRSS Ground Terminal
TPCE	telemetry processing control environment
TRS	Tape Recorder Subsystem
TSS	TDRSS Service Session
UDP	user datagram protocol
UIF	user interface
VC	virtual channel
VCDU	virtual channel data unit
VHS	VME High-Rate Subsystem
VLS	VME Low-Rate Subsystem
VLSI	very large scale integration
VME	versa module eurocard
WSC	White Sands Complex
XTE	X-Ray Timing Explorer
XTTS	XTE Test and Training Simulator