

WRDC-TR-90-8007
Volume VIII
Part 39

AD-A250 485



INTEGRATED INFORMATION SUPPORT SYSTEM (IISS)
Volume VIII - User Interface Subsystem
Part 39 - Electronic Documentation System (EDS) User's Manual

S. Barker

Control Data Corporation
Integration Technology Services
2970 Presidential Drive
Fairborn, OH 45324-6209



September 1990

Final Report for Period 1 April 1987 - 31 December 1990

Approved for Public Release; Distribution is Unlimited

MANUFACTURING TECHNOLOGY DIRECTORATE
WRIGHT RESEARCH AND DEVELOPMENT CENTER
AIR FORCE SYSTEMS COMMAND
WRIGHT-PATTERSON AIR FORCE BASE, OHIO 45433-6533



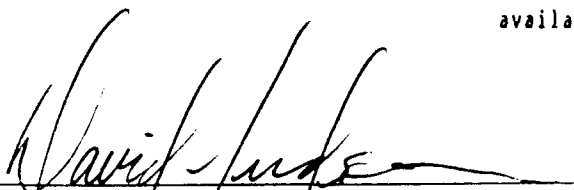
**Best
Available
Copy**

NOTICE

When Government drawings, specifications, or other data are used for any purpose other than in connection with a definitely related Government procurement operation, the United States Government thereby incurs no responsibility nor any obligation whatsoever, regardless whether or not the government may have formulated, furnished, or in any way supplied the said drawings, specifications, or other data. It should not, therefore, be construed or implied by any person, persons, or organization that the Government is licensing or conveying any rights or permission to manufacture, use, or market any patented invention that may in any way be related thereto.

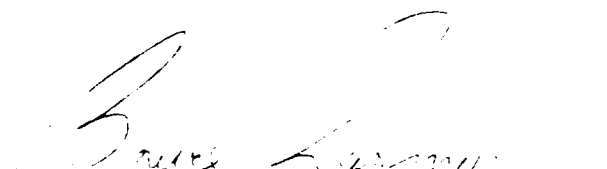
This technical report has been reviewed and is approved for publication.

This report is releasable to the National Technical Information Service (NTIS). At NTIS, it will be available to the general public, including foreign nations


DAVID L. JUDSON, Project Manager
WRDC/MTI
Wright-Patterson AFB, OH 45433-6533

25 July 91
DATE

FOR THE COMMANDER:


BRUCE A. RASMUSSEN, Chief
WRDC/MTI
Wright-Patterson AFB, OH 45433-6533

25 July 91
DATE

If your address has changed, if you wish to be removed from our mailing list, or if the addressee is no longer employed by your organization please notify WRDC/MTI, Wright-Patterson Air Force Base, OH 45433-6533 to help us maintain a current mailing list.

Copies of this report should not be returned unless return is required by security considerations, contractual obligations, or notice on a specific document.

REPORT DOCUMENTATION PAGE				
1a. REPORT SECURITY CLASSIFICATION Unclassified			1b. RESTRICTIVE MARKINGS	
2a. SECURITY CLASSIFICATION AUTHORITY			3. DISTRIBUTION/AVAILABILITY OF REPORT Approved for Public Release. Distribution is Unlimited.	
2c. DECLASSIFICATION/DOWNGRADING SCHEDULE				
4. PERFORMING ORGANIZATION REPORT NUMBER(S) UM 620344900			5. MONITORING ORGANIZATION REPORT NUMBER(S) WRDC-TR-90-8007 Vol. VIII, Part 39	
6a. NAME OF PERFORMING ORGANIZATION Control Data Corporation; Integration Technology Services		6b. OFFICE SYMBOL (if applicable) WRDC/MTI		7a. NAME OF MONITORING ORGANIZATION WRDC/MTI
6c. ADDRESS (City, State, and ZIP Code) 2970 Presidential Drive Fairborn, OH 45324-6209			7b. ADDRESS (City, State, and ZIP Code) WPAFB, OH 45433-6533	
8a. NAME OF FUNDING/SPONSORING ORGANIZATION Wright Research and Development Center, Air Force Systems Command, USAF		8b. OFFICE SYMBOL (if applicable) WRDC/MTI		9. PROCUREMENT INSTRUMENT IDENTIFICATION NUM. F33600-87-C-0464
8c. ADDRESS (City, State, and ZIP Code) Wright-Patterson AFB, Ohio 45433-6533			10. SOURCE OF FUNDING NOS.	
11. TITLE Electronic			PROGRAM ELEMENT NO. 78011F	PROJECT NO. 595600
See block 19			TASK NO. F95600	WORK UNIT NO. 20950607
12. PERSONAL AUTHOR(S) Structural Dynamics Research Corporation: Barker, S.				
13a. TYPE OF REPORT Final Report		13b. TIME COVERED 4 / 1 / 87 - 12 / 31 / 90	14. DATE OF REPORT (Yr., Mo., Day) 1990 September 30	
15. PAGE COUNT 115				
16. SUPPLEMENTARY NOTES WRDC/MTI Project Priority 6203				
17. COSATI CODES			18. SUBJECT TERMS (Continue on reverse if necessary and identify block no.)	
FIELD	GROUP	SUB GR.		
1308	0905			
19. ABSTRACT (Continue on reverse if necessary and identify block number) This document describes the integrated set of tools used to create, edit, revise, and generate manuals with well defined logical and layout structures. BLOCK 11: INTEGRATED INFORMATION SUPPORT SYSTEM Vol VIII -User Interface Subsystem Part 39 - Electronic Documentation System (EDS) User's Manual				
20. DISTRIBUTION/AVAILABILITY OF ABSTRACT UNCLASSIFIED/UNLIMITED x SAME AS RPT. DTIC USERS			21. ABSTRACT SECURITY CLASSIFICATION Unclassified	
22a. NAME OF RESPONSIBLE INDIVIDUAL David L. Judson			22b. TELEPHONE NO. (Include Area Code) (513) 255-7371	22c. OFFICE SYMBOL WRDC/MTI

FOREWORD

This technical report covers work performed under Air Force Contract F33600-87-C-0464, DAPro Project. This contract is sponsored by the Manufacturing Technology Directorate, Air Force Systems Command, Wright-Patterson Air Force Base, Ohio. It was administered under the technical direction of Mr. Bruce A. Rasmussen, Branch Chief, Integration Technology Division, Manufacturing Technology Directorate, through Mr. David L. Judson, Project Manager. The Prime Contractor was Integration Technology Services, Software Programs Division, of the Control Data Corporation, Dayton, Ohio, under the direction of Mr. W. A. Osborne. The DAPro Project Manager for Control Data Corporation was Mr. J. P. Maxwell.

The DAPro project was created to continue the development, test, and demonstration of the Integrated Information Support System (IISS). The IISS technology work comprises enhancements to IISS software and the establishment and operation of IISS test bed hardware and communications for developers and users.

The following list names the Control Data Corporation subcontractors and their contributing activities:

SUBCONTRACTOR

ROLE

Control Data Corporation

Responsible for the overall Common Data Model design development and implementation, IISS integration and test, and technology transfer of IISS.

D. Appleton Company

Responsible for providing software information services for the Common Data Model and IDEF1X integration methodology.

ONTEK

Responsible for defining and testing a representative integrated system base in Artificial Intelligence techniques to establish fitness for use.

Simpact Corporation

Responsible for Communication development.

Structural Dynamics
Research Corporation

Responsible for User Interfaces, Virtual Terminal Interface, and Network Transaction Manager design, development, implementation, and support.

Arizona State University

Responsible for test bed operations and support.

TABLE OF CONTENTS

		PAGE
SECTION	1. GENERAL	1-1
	1.1 Overview	1-1
	1.2 Purpose	1-1
SECTION	2. REFERENCES	2-1
	2.1 Reference Documents	2-1
	2.2 Terms and Abbreviations	2-3
SECTION	3. THE EDS	3-1
	3.1 Purpose of System	3-1
	3.2 Logical Structure in Documents	3-2
	3.2.1 Document Types	3-2
	3.2.2 Document Hierarchy and Logical Structure	3-3
	3.3 EDS Components	3-4
	3.4 EDS User Manual Structure	3-5
	3.5 Accessing the EDS	3-6
SECTION	4. DOCUMENT TYPE DEFINITION BUILDER (DTDBLD)	4-1
	4.1 DTDBLD	4-1
	4.2 Creating a DTD	4-3
	4.2.1 Document Hierarchy	4-6
	4.2.1.1 Search and List	4-8
	4.2.1.2 Display Document Hierarchy	4-10
	4.2.1.2.1 List Children of Element	4-10
	4.2.1.2.2 List Parent of Element	4-10
	4.2.1.2.3 Assign Attribute	4-10
	4.3 Editing a DTD	4-11
	4.4 Copy DTD	4-11
	4.5 Delete DTD	4-13
SECTION	5. LAYOUT EDITOR (EDSLE)	5-1
	5.1 Layout Requirements	5-1
	5.2 The Layout Editor	5-2
	5.2.1 Create Document Profile	5-2
	5.2.1.1 Layout Editor Function Screen	5-4
	5.2.1.1.1 Page Setup	5-5

TABLE OF CONTENTS (Continued)

		<u>PAGE</u>
	5.2.1.1.2 Generic Identifier Layout	5-7
	5.2.1.1.3 Headers/Footers	5-10
	5.2.2 Edit Document Profile	5-13
	5.2.3 Copy Document Profile	5-14
	5.2.4 Delete Document Profile	5-16
SECTION	6. SGML PARSER	6-1
	6.1 Parsing for SGML	6-1
	6.2 Running the Parser	6-1
SECTION	7. TAGGER	7-1
	7.1 Tagging a File	7-1
	7.2 Document Markup	7-1
	7.2.1 Manual Document Markup	7-2
	7.3 Automated Document Markup	7-3
	7.4 The Tagger	7-4
	7.4.1 Autotag	7-4
	7.4.1.1 State Definition	7-6
	7.4.1.2 Character Class Definition	7-8
	7.4.1.3 Pattern/Action/State Definition ...	7-10
	7.4.1.4 C Declarations	7-13
	7.4.1.4.1 Pattern Specifications Syntax	7-13
	7.4.1.5 Save Auto Tag Definition File	7-14
	7.4.2 The Lexical Writer	7-14
	7.4.3 Run Time Processor	7-14
SECTION	8. GRAPHICS SUPPORT	8-1
	8.1 EDS and Graphics	8-1
	8.2 IISS Screens Dumps	8-1
	8.3 Apple MacIntosh MacPaint Files	8-1
	8.3.1 MacPaint File	8-2
	8.3.2 MPEPSF	8-2
	8.3.3 File Transfer	8-2
	8.3.4 Print	8-3
SECTION	9. EDS FORMATTER	9-1
	9.1 Formatting the EDS Document	9-1
	9.2 EDSFMT	9-1
Appendix	A DTDBLD Document Type Definition	A-1
Appendix	B EDS Unit Test Plan Parsed Document ..	B-1
Appendix	C EDS Unit Test Plan Source Document ..	C-1
Appendix	D DECDX	D-1

LIST OF ILLUSTRATIONS

<u>Figure</u>		<u>Page</u>
3-1	IISS Logon Screen	3-7
3-2	IISS Function Screen with EDSDTD	3-7
4-1	EDS Document Type Definition Builder Screen	4-1
4-2	Input Line Command of DTD and action	4-2
4-3	Action for DTD in Tab Field	4-3
4-4	Creation of Unit Test Plan in DTDBLD Main Menu	4-5
4-5	Document Hierarchy Screen	4-6
4-6	Document Hierarchy Form	4-8
4-7	Document Hierarchy Form - Insert of Element	4-9
4-8	Document Hierarchy Form - Delete of Element	4-9
4-9	C (opy) of DTD	4-12
4-10	C (opy) of DTD Complete	4-13
4-11	D (elete) of a DTD	4-14
5-1	Layout Editor Screen	5-2
5-2	Document Profile Creation	5-3
5-3	Layout Editor Function Screen	5-4
5-4	Page Setup Screen	5-5
5-5	Generic ID Format Attribute Form	5-7
5-6	Header/Footer Form	5-10
5-7	Header/Footer Form - Define	5-11
5-8	Edit Document Profile	5-14
5-9	Layout Editor Screen - Copy	5-15
5-10	Editor Layout Screen - After Copy ..	5-16
5-11	Layout Editor Screen - Delete	5-17
5-12	Layout Editor Screen - After Delete	5-18
7-1	Generic Identifier in Markup	7-2
7-2	Autotag - IISS SGML Auto Tagging Facility	7-5
7-3	State Definition Screen - Select	7-7
7-4	State Definition Screen - Other Options	7-8

LIST OF ILLUSTRATIONS (Continued)

<u>Figure</u>		<u>Page</u>
7-5	Character Class Definition - Select	7-9
7-6	Character Class Definition - Other Options	7-10
7-7	Pattern/Action/State Defintion - Select ..	7-11
7-8	Pattern/Action/State Definition - Other Options	7-12
7-9	C Code	7-12
7-10	C Declarations Section	7-13
9-1	Document Formatter Screen	9-1

Accession For	
NTIS CRASH	↓
DTIC TAB	☐
Unannounced	☐
Justification	
By	
Distribution	
Availability Codes	
Dist	Avail. Codes Special
A-1	

SECTION 1

GENERAL

1.1 Overview

The Electronic Documentation System provides an integrated set of tools that enable the user to create, edit, revise, and generate documents with a well-defined logical structure and a well-defined layout structure.

The EDS operates in the IISS environment on those host systems which support IISS. EDS was designed and developed to accept input from and produce output for those devices which are supported within the IISS.

The EDS is a document authoring system that can be used to combine both text and graphics into a single document. In addition, EDS separates the content development from document structure and format. The EDS interfaces with the IISS via the User Interface Form Processor.

All EDS application programs which require user input use the Form Processor. The Form Processor ensures that all EDS application programs present a common forms-based user interface for the necessary user-entered input.

1.2 Purpose

The problem confronting many an author is compelling. Authors must somehow ensure that all the various requirements for a document are met correctly in the time allowed. The EDS was developed to assist the author in this effort.

EDS allows for the separation of the author from the format or layout requirements of a document. The author is separated from the document structure considerations. Presently, many technical documents are developed by an author who must be concerned with content, format, and structure.

EDS: These are the steps in the creation of a document using

1. Define the document type.
2. Define logical structure of the document type.
3. Define the document layout style.
4. Develop the document content
 - A. Markup or tag the document content in English-like commands, i.e. SGML
 - B. Author can include any special requirements as defined in the layout style.
 - C. Document produced in VAX WPS can be run through the EDS Tagger to strip out WPS control characters and replace with document type information.
5. Completed document content is parsed
 - A. This would include the WPS document which has been run through the Tagger.
6. Document is ready for the Formatter
 - A. External graphics files are run through the appropriate conversion program for inclusion into the final form document.
 - B. The parsed document content, the output of Step 5 is ready for the Formatter.
 - C. The graphics and the parsed document are combined in this step for the final output form of the document.

The above outlined steps are the overall steps in the development of an EDS document. Not all document developers, especially authors of document content, are required to follow these steps. The EDS was produced to separate these steps or functions for document developers. The author of a document would be independent of many of the above steps.

SECTION 2

REFERENCES

2.1 Reference Documents

- [1] IDS150120000C, ICAM Documentation Standards, SYSTRAN, 15 September 1983.
- [2] DS 620344700, Form Processor Development Specification, Structural Dynamics Research Corporation, 31 May 1988.
- [3] OM 620344000, Terminal Operator Guide, Structural Dynamics Research Corporation, 31 May 1988.
- [4] UM 620344200, Form Processor User Manual, Structural Dynamics Research Corporation, 31 May 1988.
- [5] DS 620344900, Electronic Documentation System (EDS) Development Specification, Structural Dynamics Research Corporation, 31 May 1988.
- [6] UTP620344910, Electronic Documentation System (EDS) SGML Parser Unit Test Plan, Structural Dynamics Research Corporation, 31 May 1988.
- [7] UTP620344920, Electronic Documentation System (EDS) Tagger Unit Test Plan, Structural Dynamics Research Corporation, 31 May 1988.
- [8] UTP620344930, Electronic Documentation System (EDS) Document Type Definition Builder Unit Test Plan, Structural Dynamics Research Corporation, 31 May 1988.
- [9] UTP620344940, Electronic Documentation System (EDS) Layout Editor Unit Test Plan, Structural Dynamics Research Corporation, 31 May 1988.

- [10] UTP620344950, Electronic Documentation System (EDS) Formatter Unit Test Plan, Structural Dynamics Research Corporation, 31 May 1988.
- [11] UTP620344960, Electronic Documentation System (EDS) MacPaint to Postscript EPSF Translator Unit Test Plan, Structural Dynamics Research Corporation, 31 May 1988.
- [12] American National Standard for Information Systems - Computer Graphics - Metafile for the Storage and Transfer of Picture Description Information, ANSI X/3.122-1986, American National Standards Institute, August 27, 1987.
- [13] Information Processing - Text and Office Systems - Standard Generalized Markup Language (SGML), ISO 8879, International Organization for Standardization, 15 October 1985.
- [14] LEX - Lexical Analyzer Generator, IS Workbench for VAX/VMS Programmers Guide, M. E. Lesk.
- [15] Office Document Architecture/Office Document Interchange Format, ISO/DP8613/1-6, International Organization for Standardization, October, 1985 (Draft).
- [16] POSTSCRIPT Language Reference Manual, Adobe Systems Incorporated, Addison-Wesley Publishing, Inc., October, 1986.

2.2 Terms and Abbreviations

Attribute: A characteristic used to qualify an element within a document.

Character Set: A mapping of a character repertoire onto a code set such that each character is associated with its coded representation.

Compound Document: A document which may contain mixed content i.e. text, graphics, etc.

Conforming SGML Application: An SGML application that requires documents to conform to SGML documents, and whose documentation meets the requirements of this International Standard.

Conforming SGML Document: An SGML document that complies with all provisions of ISO 8879.

Cursor Position: The position of the cursor after any command is issued or in a certain location on the computer screen display.

Descriptive Markup: Information added to a document which enables an application program to process the document.

Element: A component of the hierarchical structure defined by a document type definition; it is identified in a document instance by descriptive markup, usually a start-tag and end-tag.

Entity: A collection of characters that can be referenced as a unit.

Field: A two-dimensional space on a terminal screen.

Form: A structured view, presented on a computer screen display, which may be imposed on windows or other forms. A form is composed of fields, these fields may be defined as forms, items, or windows.

IISS: Integrated Information Support System, a test computing environment used to investigate, demonstrate, and test the concepts of information management and information integration in the context of Aerospace Manufacturing. The IISS addresses the problems of

integration of data resident of heterogeneous data bases supported by heterogeneous computers interconnected via a Local Area Network.

IISS Function Screen: The first screen that is displayed after logon. It allows the user to specify the function to access and the device type and device name on which to work.

Menu: A screen display which presents user selectable functions or actions to be taken.

Message: Descriptive text which may be returned in the message line on the terminal screen. Messages are used to warn of errors or provide additional information.

Message Line: A line on the terminal screen that is used to display messages.

ODA: Office Document Architecture, a standard which supports the interchange of electronic compound documents in such a way as to allow their imaging, processing, or reformatting.

ODIF: Office Document Interchange Format, a standard for encoding document structures defined by the ODA which would enable the exchange of compound documents between systems operating within a MAP/TOP environment.

SGML: Standard Generalized Markup Language, a language for describing document structures, consisting of descriptive markup which is added to a document to indicate where logical elements such as sections and paragraphs begin and end.

Tag: Descriptive markup indicating the start or end of a logical element.

Window: Dynamic area of a terminal screen on which predefined forms may be placed at run time.

SECTION 3

THE EDS

3.1 Purpose of System

The EDS is a system for document processing. Document processing is accomplished by various methods. These methods include typesetting, typewriters, word processing systems, desktop publishing systems, and other specialized packages. Text processing has been recognized as an important phase in the development and the delivery of any product.

The selected text processing standard for the EDS is the Standard Generalized Markup Language (SGML). SGML is an International Standards Organization (ISO) standard. This standard is a descriptive high level language which structures and formats text. The formatting of text is that set of commands that physically presents the text on the printed page.

There are other languages that format and structure documents. These are usually procedural markup languages. Procedural markup languages embed the format and structure requirements in the document text. This can be seen in most word processing systems.

Physical presentation of the text includes the structure of the text and the format of the text. There are special processing techniques for the structure of the text. These structural requirements, for example, may include:

- the commands for a section title
- a paragraph heading
- a list
- a figure or illustration
- a glossary
- an appendix
- an index

The format of the text includes special requirements also. Format requirements may include:

- the page number location
- the use of a top notation on the page (a header)

- the use of a bottom notation on the page (a footer)
- the margin setting for the text
- the numbers of lines to a page
- the size (pitch) of the characters
- the style (font) of the characters

3.2 Logical Structure in Documents

A book is a compilation of information on a subject, whether that subject be fiction or fact. A book that is purchased from a bookstore has a defined structure to it. This structure can be defined as: a front cover, a copyright and trademark page, a title page, a table of contents, subsequent chapters, subsequent paragraphs in those chapters, an index, and a back cover. There is implied in this structure a certain generic sequence. The front cover to the table of contents can be considered the "front matter" of the book. The chapters and paragraphs can be considered the "body". The index and back cover can be considered "back matter".

The technical document structure can be expressed similarly, for example, an ICAM document type. Let's define the structure of a unit test plan: the title page, the notice page, the government distribution page, the preface, the table of contents, the list of illustrations, the major sections, paragraph headings with numbers, paragraphs, appendices (as required), an index, and perhaps a glossary. The title page to the list of illustrations would be the front matter. The sections to the appendices would be the body. The index and the glossary would be the back matter.

As you can see from the two structures that we have defined there is a starting point (the front matter), a middle (the body), and an ending point (the back matter) for the book and the technical document. This can be termed logical structure. A document structure must be clearly understood when defining a document type in the EDS.

3.2.1 Document Types

Now that we have been able to understand a high level document structure, we must begin to understand the lower levels of the document structure. We are going to analyze the additional elements of the logical structure of a document.

Logical document structure can be expressed as an hierarchy of parents, children, and groups of parents and children. The document structure is composed of a high level or generic component that has lower level elements. These lower level elements may be composed of other lower level elements also. This is how the terms parent (high level element) and child (low level element) have evolved. Without the parent there would be no child, a child may be a parent to another child, and so on. A parent or a child may have siblings also. A sibling to front matter, for example, would be body.

As an EDSDTD user, you will define the hierarchy of a document in the expression of the relationships between the parents and children within a document. These logical elements will also be described to the system by the number of occurrences for each element.

In addition to the hierarchical structure of the document type, there may be descriptors to the elements of the structure. These descriptors are called attributes. An attribute could further qualify an element. As an example, the logical structure element book may have a status attribute associated with it which defines the book to be a draft or final copy. Element and attribute statements are the basis of the definition of a document type logical structure.

3.2.2 Document Hierarchy and Logical Structure

The creation of a document involves analyzing the document type to be created. Document analysis is accomplished, by many authors, by the use of an outline of the basic structure of the document. This outline can be used to illustrate the hierarchy of the logical structure of the document. Let's analyze and outline an unit test plan as our test example.

The unit test plan would follow those general requirements as documented in the ICAM Documentation Standard. The generic logical structure of this unit test plan would be front matter, body, and rear matter. Let's briefly outline this document:

unit test plan

- Front Matter
 - Cover Page
 - Notice Page
 - Government Distribution Page
 - Preface
 - Table of Contents
 - List of Illustrations
 - List of Tables

Body

- Section One
- Section Two
- Section Three
- Section Four
- Appendix A

Rear Matter

- Index
- Glossary

The outline could be further developed to include those elements of each of these level elements that we listed. For example, Cover Page could be further described:

Cover Page

- Title of Project
- Title of Configuration Item
- Title of Computer Program
- Date
- Address of Developer
 - Address line 1
 - Address line 2
 - Address line 3
 - Address line 4

As you can see from this further breakdown of the element Cover Page, there are many other elements that could be defined.

3.3 EDS Components

EDS is composed of tools that enable a user or group of users to create, edit, revise, and generate documents. These documents may span all types of generated text from a simple office memo to a complex specification. Most importantly, EDS will support the compound document which combines text and graphics together.

The various components of EDS support certain functions of the system. Those functions include the definition of a document type, the layout or format definition, the conformance to the document type, the conversion from the WPS word processing package to SGML, the inclusion of graphics from external sources, and the formatting of the document for print.

The following lists the functions and each component for the function.

1. The definition of document type is found in the DTDBLD.
2. The layout or format definition is found in the Layout Editor.
3. The conformance to document type is found in the SGML Parser.
4. The conversion from word processing package is found in the Tagger.
5. External graphics conversion and inclusion into document is found in the Graphics section.
6. Formatting the document for print is found in the Formatter.

3.4 EDS User Manual Structure

This unit test plan is structured to give the general user the information necessary to function properly in EDS. There are several steps that must be followed to ensure that the system will give optimum performance to the user. The structure of this manual assists in familiarizing the user with the concepts of logical structure, layout structure, generic ids or macros, and other text processing terminology.

The following sections discuss the individual components of EDS. Each component has specific examples of text processing concepts as necessary. The Formatter, for example, requires some information from the user. However, this is specific information with no real text processing concepts to understand.

Remaining sections of this manual are organized as follows:

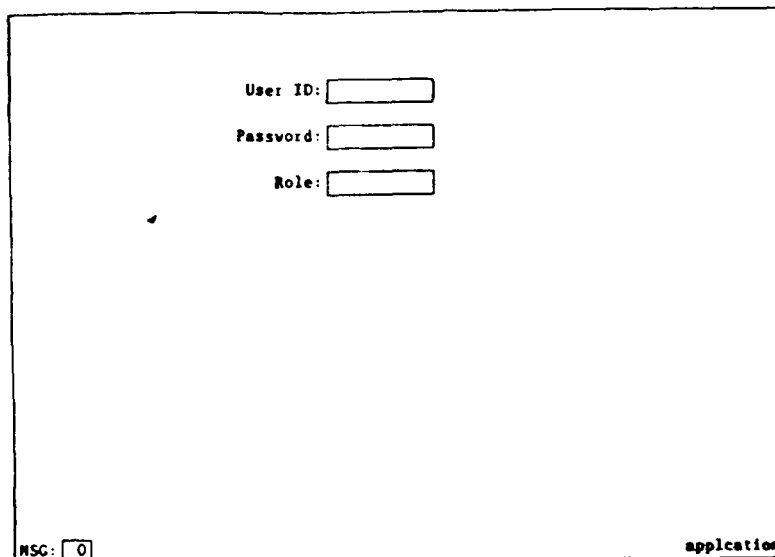
- Section 4 - DTDBLD, build of document types
- Section 5 - Layout Editor, definition of layout
- Section 6 - SGML Parser, parsing document for SGML
- Section 7 - Tagger, conversion of word processing file
- Section 8 - Graphics in the EDS
- Section 9 - the Formatter, production of final form

Each area of discussion includes component error messages, functions, help tools, and key sequences to be typed in.

As we progress through these areas a sample document is used to illustrate the logical structure, the layout structure, and the graphics for the document. An example of document source text can be found in Appendix B.

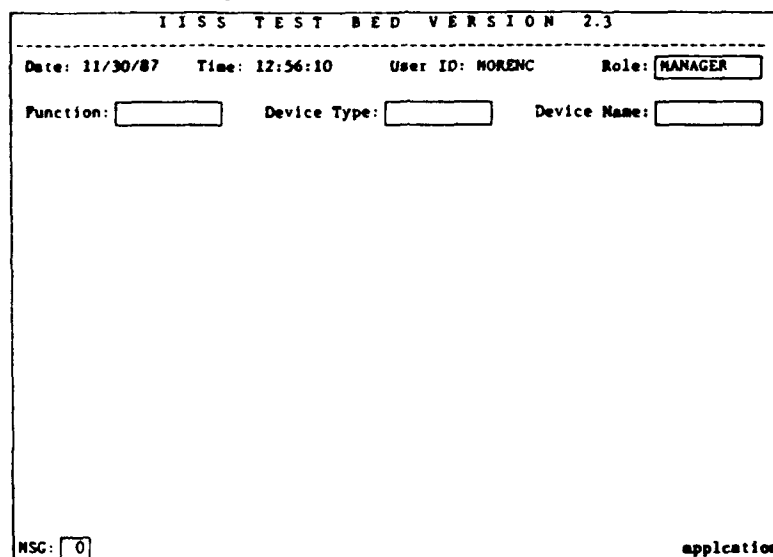
3.5 Accessing the EDS

There are two methods of accessing the EDS. EDS was developed as an IISS tool and is therefore accessed from the IISS Function Screen. The EDS, including all the components that comprise the system, may be accessed through the IISS.



A screenshot of the IISS Logon Screen. It features three input fields for 'User ID:', 'Password:', and 'Role:'. At the bottom left, there is a status bar with 'MSG: 0'. At the bottom right, the word 'application' is displayed.

Figure 3-1 IISS Logon Screen



A screenshot of the IISS Function Screen with EDSDTD. The title bar reads 'IISS TEST BED VERSION 2.3'. Below the title bar, there is a dashed line. Below the dashed line, there is a status bar with 'Date: 11/30/87', 'Time: 12:56:10', 'User ID: MORENC', and 'Role: MANAGER'. Below the status bar, there are three input fields for 'Function:', 'Device Type:', and 'Device Name:'. At the bottom left, there is a status bar with 'MSG: 0'. At the bottom right, the word 'application' is displayed.

Figure 3-2 IISS Function Screen with EDSDTD

SECTION 4

DOCUMENT TYPE DEFINITION BUILDER (DTDBLD)

This section is intended to give the user the information necessary to begin building documents in the EDS. An understanding of document types and how to define them is of utmost importance for the user of EDS. Additionally, the ability to understand the logical structure of a document is a prerequisite to the DTDBLD function. These concepts are discussed in Section 3.2 and 3.4 respectively.

4.1 DTDBLD

This is the program module of the EDS that defines the document type and the document structure. The logical structure of the document will be defined in the DTDBLD program. This program can be accessed by logging into the IISS, entering EDSDTD at the "Function" input prompt, and pressing the <ENTER> key.

After the previous screen, Figure 3-2, the EDS Document Type Definition Builder screen will appear. A list of existing document type definitions will appear in the form. This list is scrollable and can be scrolled using the <SCROLL/PAGE> mode key defined in the IISS Terminal Operator Guide.

Action	DTD Name	Descriptor
	DEMODOC	
	ICAVUTP	
	ICAVDOC	
	EDSUTP	
	EDSUTD	
	DTDBLD	

Figure 4-1 EDS Document Type Definition Builder Screen

There are only four actions that can be performed on a DTD. These actions are:

1. E (dit)
2. C (opy)
3. D (elete)
4. S (earch)

These actions can be entered into the form in two different ways. The action can be entered directly on the first input line of the form along with the DTD name. Completion of the action will occur after the <ENTER> key is pressed.

EDS - Document Type Definition Builder 1/22/88 13:50:51

DTD Name ☒ Action (E, D, C, S)

Existing DTD Directory Entries

Action	DTD Name	Description
	DENODOC	
	ICANUTP	
	ICANDOC	
	EDSUTP	
	EDSUTD	
	DTDBLD	

MSG: application

Figure 4-2 Input Line Command of DTD and action

The second method for entering the DTD action is through cursor movement. The cursor is moved via the <TAB> key to the desired DTD and the command is entered in the action field on the form. This action is then processed by pressing the <ENTER> key.

EDS - Document Type Definition Builder 1/22/88 13:50:51

DTD Name ☒ Action (E, D, C, S)

Existing DTD Directory Entries

Action	DTD Name	Description
☐	DEMODOC	
☐	ICAMUTP	
☐	ICAMDOC	
☐	EDSUTP	
☐	EDSDTD	
☐	DTDBLD	

MSG: application

Figure 4-3 Action for DTD in Tab Field

The creation of a new DTD is accomplished by entering the DTD name, the E (dit) action command, and pressing <ENTER>. The modification to an existing DTD would be accomplished by using the E (dit) action in the selected DTD action field and pressing <ENTER>. The C (opy) command is entered in this same fashion.

4.2 Creating a DTD

The creation of a DTD involves analyzing the document type to be defined. This can be done by thinking of the document that you wish to create in outline form. The outline can be used to illustrate the hierarchy of the logical structure of the document. Let's analyze and outline an unit test plan as our test example.

The unit test plan would follow those general requirements as documented in the ICAM Documentation Standard. The document that we outlined in Section 3.4.1 gives us the hierarchical or logical structure of a document to create in the DTD. Let's review the structure we outlined in the previous section. This outline included the generic type of document, a unit test plan; a generic beginning for the document, front matter; a

generic content area for the document subject matter, a body;
and an area for additional reference material on the document, a
rear or back matter.

unit test plan

Front Matter

- Cover Page
- Notice Page
- Government Distribution Page
- Preface
- Table of Contents
- List of Illustrations
- List of Tables

Body

- Section One
- Section Two
- Section Three
- Section Four
- Appendix A

Rear Matter

- Index
- Glossary

This outline illustrates the document type level 0, unit
test plan; the generic level 1, i.e. front matter; and the next
level 2, i.e. Cover Page, for the document structure. The
outline could be further developed to include those elements of
each of these level elements that we listed. For example, Cover
Page could be further described:

Cover Page

- Title of Project
- Title of Configuration Item
- Title of Computer Program
- Date
- Address of Developer
 - Address line 1
 - Address line 2
 - Address line 3
 - Address line 4

As you can see from this further breakdown of the element Cover
Page, there are many other elements that could be defined.

We took the level 2 element Cover Page and defined the next
level down from it. This level 3 element, Address of Developer,
had children defined for it. These children of level 3 element
Address of Developer became level 4 elements.

DTDBLD will assist the user in defining this hierarchical
structure to each document type defined. The user must
understand the parent-child relationship when defining this
structure. The DTDBLD is considered a tool for those
knowledgeable in document design, structure, and analysis. This
is especially true in the creation of a document type
definition.

The required input for you, the user, to create a new document type definition and to define that logical structure is the E (dit) action code on the DTDBLD main menu or screen display. To create the document type, unit test plan, the required inputs would be an E in the action code field, the name of the document type in the DTDname field, and the pressing of the <ENTER> key. Figure 4-6 illustrates the inputs for this step.

EDS - Document Type Definition Builder 1/22/88 15:50:51

DTD Name Action (E, D, C, S) ☒

Existing DTD Directory Entries

Action	DTD Name	Description
	DEMODUC	
	ICAMUTP	
	ICAMDOC	
	EDSUTP	
	EDSOTD	
	DTDBLD	

MSG application

Figure 4-4 Creation of Unit Test Plan DTD in DTDBLD Main Menu

The EDS will go to the next screen form for the definition of the document type. This screen will resemble the following figure.

Document Hierarchy						
Level	Common Name	Generic Identifier	Total Instances	Order of Children	Group may Occur	Instance may occur
0				SEO	1	

MSG: 1 Enter/complete definition for root level element application

Figure 4-5 Document Hierarchy Screen

4.2.1 Document Hierarchy

The only input area on the Document Hierarchy Screen is that for the DTD itself (remember level 0 from our outline). It is up to the user to now describe and define the rest of the document logical structure. Again, our outline should come in handy for this definition.

The level 1 elements should be entered into the form. Those elements (again from our outline) would be front matter, body, and rear matter. The form input areas to be filled in include the level, the common name, the generic identifier, the total instances of the element, the order of children, how often the group may occur, and how often the instance of the group may occur. These input fields have significant impact to the document structure. The input fields and meanings follow.

Level: The number of the element in the document hierarchy, beginning the document type at level 0.

Common Name: The name associated with the element, not one that is an abbreviation or a devised term for the element.

Generic Identifier: This would be a term devised for the element or an abbreviation.

Total Instances: This is system generated.

Order of Children: This is a necessary input that will tell the EDS how to process the children of a given element.

The system accepts only the following for this input field:

SEQ: All elements will occur in required order, i.e. sequential order

NSEQ: All elements will occur, but in no special or required order, i.e. non-sequential order

OR: One and only one of the children will be used

NONE: There are no children to this parent

Group May Occur: Defines how often an element may occur, possible values are:
(Children)

Z1: Zero or one time

ZM: Zero or many times

1M: One or many times

1: Once

Instance May Occur: Defines how often that instance of the element may occur, possible values are the same as that of above Group (Z1, ZM, 1M, 1)

The actions performed in the Document Hierarchy Form are Search and List document type elements, Insert elements, and Delete elements.

4.2.1.1 Search and List

This is the default action of the Document Hierarchy Form. When this form is accessed from the DTDBLD Main Menu, it automatically does a search for the elements and a list of the elements that compose the document type. For a newly created document type there will be no elements except that of the document type itself, level 0.

Level	Common Name	Generic Identifier	Total Instances	Order of Children	Group may Occur	Instance may occur
0				SEO	1	

MSG: 1 Enter/complete definition for root level element application

Figure 4-6 Document Hierarchy Form

Document Hierarchy						
Level	Common Name	Generic Identifier	Total Instances	Order of Children	Group may Occur	Instance may occur
0	user manual	um		SEQ	1	1
1	front matter	front				
1	body matter	body		seq	1a	
1	rear matter	rear				

MSG: 1 Enter/complete definition for root level element application

Figure 4-7 Document Hierarchy Form Insert of Element

Document Hierarchy						
Level	Common Name	Generic Identifier	Total Instances	Order of Children	Group may Occur	Instance may occur
0	user manual	um	1	SEQ	1	1
1	front matter	front	1	NONE	1	1
1	body matter	body	1	SEQ	1a	1

MSG: 0 application

Figure 4-8 Document Hierarchy Form Delete of Element

4.2.1.2 Display Document Hierarchy

The elements that you have entered into the Document Hierarchy form have defined the document type and logical structure to EDS. The ability to view this logical structure at the element level is one that will be used for analysis of the document. This will be especially true for verification of the document logic after initial definition of the document structure.

There are several functions that exist in this form for the verification process. These functions are to list the children or subordinates of an element, to display the parent of the element with the parent's subordinates, and to assign attributes to an element. A description of these functions follows, however, examples of the functions and commands can be found in Section 4.4.2 Edit a DTD.

4.2.1.2.1 List Children of Element

This action within the Display Document Hierarchy Menu is the default action of the form. The action taken in the previous form, see Figure 4-10, automatically accesses this form with the element and the list of children for that element.

4.2.1.2.2 List Parent of Element

This action within the Display Document Hierarchy Menu is used to list the parent of the element. This screen list will display the element's parent and the subordinates of the parent. Included in the subordinates displayed on the screen should be the element.

This action is accomplished by placing the cursor on the input line at the top of the form. The cursor should be on the Generic Identifier position. By pressing the <PF13> key on your keyboard, the form will change to that of the parent of the element. Reference Section 4.4.2 Edit a DTD for examples of the commands and usage of this form.

4.2.1.2.3 Assign Attribute

This action within the Display Document Hierarchy Menu is used to assign attributes to children of an element. Place the cursor on the child of the element that you wish to assign an

attribute to and press the <PF15> key. The <PF15> key is the Attribute key and brings in the Attribute Form of the Display Hierarchy Form. There are two modes in this form, Insert and Delete.

The Insert mode is the default mode for this form. To insert an attribute to a child element, place the cursor on an empty Attribute Input Field. The attribute should be entered into this input field and the <ENTER> key should then be pressed. The next input area on the form is the Default Field. The cursor should be placed on this field and the data typed in along with the <ENTER> key. The next input area is the Associated Values Field on the form. The cursor should be moved to that input field and the data entered with an <ENTER>. For examples of these commands, see Section 4.4.2 Edit a DTD.

4.3 Editing a DTD

The required input for you, the user, to create a new document type definition and to define that logical structure is the E (dit) action code on the DTDBLD main menu or screen display. There is no difference between the create document type and the edit of document type. To create the document type, dtdbld, the required inputs would be an E in the action code field, the name of the document type in the DTDname field, and the pressing of the <ENTER> key. Figure 4-6 illustrates the inputs for this step. To edit that same DTD, you would enter the DTD name and the action code E (dit) on the first input line of the initial screen and then press <RETURN> or you would move the cursor to the action field next to the DTD name on the screen display and press <RETURN>.

4.4 Copy DTD

The C (opy) command is an action to be used when you wish to create a similar document type to an existing one. This can be beneficial in that you would not have to create a new document type from the very beginning, but could edit a copy of a similar document type. Another use of the C (opy) action is for maintaining the previous copy of a revised document type. The C (opy) action is entered in the initial EDSDTD screen.

```
EDS - Document Type Definition Builder      12/15/87  9:37:36

      DTD Name  Action (E, D, C, S)

      Existing DTD Directory Entries

Action  DTD Name      Description
☐ DTDBLD

From: DTDBLD  To: DTDSAV  Description= EST
MSG: ☐ application
```

Figure 4-9 C (opy) of DTD

A message will appear at the bottom of the screen. This message will require input from you. The first prompt will be for the name of the new file and the second prompt will be the description of the new DTD to be copied into. After entering these two inputs, press the <ENTER> key.

The C (opy) action will have been performed, resulting in the following form.

```
EDS - Document Type Definition Builder      12/15/87 12:26:36

      DTD Name [REDACTED]  Action (E, D, C, S)

      Existing DTD Directory Entries

Action:  DTD Name      Description
        [REDACTED]    DTDSAV      TEST
        [REDACTED]    DTDBLD

MSG: [REDACTED] application
```

Figure 4-10 C (opy) DTD complete

4.5 Delete DTD

The D (elete) action is performed similarly to the C (opy) action. The action is placed in the action field for the appropriate DTD and the <ENTER> key is pressed. D (elete) may also be performed from the DTD command line. The action is entered on that line along with the DTD name and the <ENTER> key.

```
EDS - Document Type Definition Builder      12/15/87 12:26:36

      DTD Name  Action (E, D, C, S)

      Existing DTD Directory Entries

Action  DTD Name      Description
 DTDSAV      TEST
         DTDBLD

      OK to delete Document Type Definition? DTDSAV ?      YES NO
MSG:  application
```

Figure 4-11 D (elete) of a DTD

The system will verify that the selected DTD is indeed one to be deleted from the system. You must enter a positive response to the question that will appear if you wish to delete the DTD. The displayed message will appear at the bottom of your screen. You must respond to this message:

Okay to delete Document Type Definition? DTDname Yes No

The default response will be "YES" and can be entered by pressing the <ENTER> key. The screen display will come back without the deleted DTD. To not delete the selected DTD place the cursor on the "NO" and press the <ENTER> key. The DTD will not be deleted and the screen will be re-displayed with no action taken.

SECTION 5

LAYOUT EDITOR (EDSLE)

This section will explain layout structure and any additional layout tools for use in the EDS. Layout or format structure is the physical placement of the document text on the page. In any document there are certain layout requirements that are standard. These layout or format requirements range from the page margins to the space between lines on the page.

The EDS Layout Editor is a forms based program that enables you to define the layout style for a document type. The layout style is defined with SGML tags within a document. The DTD for the document must be defined before the layout style can be defined. The layout style (formatting parameters) that you will define for the DTD will be the Document Profile.

5.1 Layout Requirements

The Document Profile is the set of formatting commands that you will give the Layout Editor for the DTD. This file will contain the layout style for the logical elements of the DTD and a set of global formatting attributes for the document. The DTDBLD component of EDS builds the structure of the document including the element name. The Document Profile component of EDS builds the formatting or layout style parameters for the document elements.

Global formatting attributes that you will enter into the Layout Editor will range from the page margin requirements to requirements for the text character size and style. Additionally, the layout style for the logical elements of the document type can be defined in the creation of macros. Macros are a set of layout commands for a specific document element.

EDS allows you to create macros in the Layout Editor. These macros are the format or layout requirements which relate back to a specific generic identifier. For example, a figure in a document must be processed in specific logical steps. The document element Figure must have a certain amount of space reserved for it, it must have a beginning place, and an ending place; the document element Figure Number must be centered on the page along with the figure caption. The set of commands that process the format of the document element Figure, are termed macros.

5.2 The Layout Editor

The Layout Editor is accessed through the IISS function menu. You must log into the IISS with a valid user id and account. The IISS function screen should appear, you must enter EDSLE in the Function field and press <ENTER>.

The next screen, or form, to appear is the initial Layout Editor Screen. This screen will display in a list format the previously defined DTDs and any previously defined Document Profiles. There may be several Document Profiles to a single DTD. These multiple profiles can be used with the same document type and document text to process the document in various layout or format styles. The action codes that are available in this form are E (dit), C (opy), and D (elete).

IISS Electronic Documentation System (EDS) 12/ 4/87 14:53:26

Layout Editor

Action	DTD Name	Document Profile	Description
☐			
☐	DENODOC	2DEMO	SECOND DEMO DOCUMENT
☐	DENODOC	DENODOC	
☐	EDSUTP	EDSLE1	eds layout editor utp
☐	ICAMUTP	ICAMUTP	ICAM Unit Test Plan

MSG: application

Figure 5-1 Layout Editor Screen

5.2.1 Create Document Profile

To create a Document Profile for the DTD Unit Test Plan that was created in Section 4.4, the E (dit) action code must be entered in the Action Field on this form. This Action Field is located to the left of the DTD name. The Document Profile name must be entered also. A Description Field is available for your use. This Description Field can be used at this time for any description that you wish to record with the Document Profile. It should be noted, this field can be modified at any time.

Let's define a Document Profile to the DTD Unit Test Plan that you defined in Section 4.4. Place the cursor on the Action Field where the DTD Unit Test Plan is listed and enter an 'E'. Tab the cursor over to the Document Profile Field and enter the profile name, 'ICAMUTP'. Tab the cursor to the Description Field on the form and enter in a description of the profile. You might enter in the following:

ICAM unit test plan manual 12/87

Now press the <ENTER> key for the system to process your creation of the Document Profile ICAMUTP for the DTD Unit Test Plan.

IISS Electronic Documentation System (EDS) 12/ 4/87 14:53:26

Layout Editor

Action	DTD Name	Document Profile	Description
☐			
☒	DEMODOC	2DEMO	SECOND DEMO DOCUMENT
	DEMODOC	DEMODOC	
	EDSUTP	EDSLE1	eds layout editor utp
	ICAMUTP	ICAMUTP	ICAM Unit Test Plan

MSG: 0 application

Figure 5-2 Document Profile Creation

The Document Profile is created. The system will now proceed to the Layout Editor functions for the selected Document Profile.

5.2.1.1 Layout Editor Function Screen

This screen form will be the one from which you select the Layout Editor function to perform. There are three functions to be performed. They are Page Setup, Generic ID Layout, and Headers/Footers.

```
IISS Electronic Documentation System (Layout Editor) 12/ 4/87 15:07:34

Menu

Page Setup
Generic ID Layout
Headers/Footers

MSG: 2 application
```

Figure 5-3 Layout Editor Function Screen

The three functions that are performed from this screen form are the functions necessary to define the layout or format of the document, i.e. the Document Profile. The following is a brief description of these functions.

Page Setup: Definition of top and bottom margins, left and right margins on odd and even pages, line spacing, leading spaces, font size (pitch), and font name (type).

Generic ID Layout: Definition of format parameters for the elements of a document, i.e. build of relationship between a generic id and its layout style

Headers/Footers: Definition of the reserved area at the top and the bottom of the page of a document for additional document information.

The functions are selected from the form by placing the cursor on the function, use the <TAB> key, and pressing the <ENTER> key. Each function selected goes into a subordinate form. Let's begin with Page Setup.

IJSS Electronic Documentation System (Layout Editor) 12/ 4/87 15:07:45

Page Setup for Document Profile DENODOC

Margins		Line Spacing
Top	1.00 IN	1.00
Bottom	1.00 IN	Leading 14
(Even) Left	1.00 IN	Font Size 12
Right	1.00 IN	Font Name Times-Roman
(Odd) Left	1.00 IN	
Right	1.00 IN	

MSG: 0 application

Figure 5-4 Page Setup Screen

5.2.1.1.1 Page Setup

The Page Setup function is the function that you will use for the margins, line spacing, leading spaces, and font characteristics. The Screen displays the Document Profile that you are editing and the input fields for the Page Setup Function. Default assignments are already in place for these input fields. Changes can be made to each input field by moving the cursor to the field and typing in the desired change. The input fields are:

Top Margin: This parameter is used to reserve space at the top of the page for a margin. Its value can be expressed in inches, centimeters, characters, etc.

Bottom Margin: This parameter is used to reserve space at the bottom of the page for a margin. Its value can also be expressed in inches, centimeters, characters, etc.

Even Left Margin: This parameter is the left margin space for an even numbered page. Its value can be expressed the same as the other margins.

Even Right Margin: This parameter is the right margin space reserved for an even numbered page. Its value can be expressed the same as the other margins.

Odd Left Margin: This parameter is for the reserved left margin space for an odd numbered page. Its value can be expressed the same as the other margins.

Odd Right Margin: This parameter is for the reserved right margin space for an odd numbered page. Its value can be expressed the same as the other margins.

Line Spacing: This value is usually expressed in inches. A value of 2.00 would be two lines or double spaced.

Leading: This value is expressed in the amount of space between a characters printed on the page.

Font Size: This value is expressed in the size or pitch of a character, i.e. 10, 12, 14.

Font Name: This value is expressed in a name of a font type or style. The supported fonts for the EDS are maintained for you. The <HELP> key will produce a list of available fonts while on this input field.

After the changes have been made to the input fields, press the <ENTER> key to save the changes. The System will redisplay the screen with the new values.

In order to go on to the next function, you must press the IISS QUIT key, <PF4>, this will take you back to the Layout Editor Function Screen. You can now select another function by placing the cursor on the selected function and pressing the <ENTER> key. Let's go to the Generic ID Layout Screen.

5.2.1.1.2 Generic Identifier Layout

The Generic ID Layout Screen displays the DTD and the Document Profile at the top of the form. This screen also displays the Generic Identifier that was defined in the DTD and the macro name of that Generic Identifier. You must select a macro from the displayed list to change the layout style of that macro. Place the cursor on a macro name in the listed area on the form and press <ENTER>. This action will take us to the Format Attribute Form for the selected macro. If no action is desired, the IISS QUIT key, <PF4>, would quit this form and go back to the Layout Editor Function Screen.

IISS Electronic Documentation System (Layout Editor) 12/ 4/87 15:08:04

DTD : DEMODOC
Document Profile : DEMODOC

Fully Qualified Generic Identifier

Fully Qualified Generic Identifier	Macro Name
DEMODOC	\$DEMODOC
DEMODOC.BODY	\$BODY
DEMODOC.BODY.LIST	\$LIST
DEMODOC.BODY.LIST.ITEM	\$ITEM
DEMODOC.BODY.FIGURE	\$FIGURE
DEMODOC.BODY.FIGURE.FIGDESC	\$FIGDESC
DEMODOC.BODY.FIGURE.FIGDESC.FIGTITLE	\$FIGTIT
DEMODOC.BODY.FIGURE.FIGDESC.FIGRUM	\$FIGRUM
DEMODOC.BODY.FIGURE.FIGDESC.FIGREF	\$FIGREF
DEMODOC.BODY.FIGURE.REPORT	\$REPORT
DEMODOC.BODY.FIGURE.GRAPHIC	\$FIGTIT
DEMODOC.BODY.FIGURE.FIGBODY	\$FIGBOC

MSC More application

Figure 5-5 Generic Id Format Attribute Form

This form has no default values assigned to it. The input fields on the form must be tabbed to using the <TAB> key and enter the data. Input field values vary on this form:

- Bold:** This is used to define the Generic ID as boldface type on the page. An 'x' indicates the field is selected. A blank in the field is considered an 'N'.
- Underline:** This is used to define the Generic ID as underlined type on the page. This has the same values as Bold.
- Reverse:** This is used to define the Generic ID as a reversed print on the page, i.e. print is white on black. An 'x' indicates the field is selected, a blanks indicates not selection.
- Italics:** This is used to define the print style of the Generic ID as italic. Values are a 'x' for yes or blank for no.
- No Break:** This is used to define whether or not the GI is to span multiple pages. Values are a 'x' for yes or blank for no.

Literal: This is used to define whether or not the Generic ID is a literal. Values are 'x' for yes or blank for no.

No Advance: This is used to define whether or not the Generic ID causes an advance line after being processed. Values are 'x' for yes or blank for no.

Justify Center: This is used to define whether or not the Generic ID is justified to the center of the defined page. Values are 'x' for yes or blank for no.

Justify Right: This is used to define whether or not the Generic ID is justified to the right of the defined page. Values are 'x' for yes or blank for no.

Justify Full: This is used to define whether or not the Generic ID is justified between the left and right margins of the defined page. Values are 'x' for yes or blank for no.

Indent Left: This is used to define the Generic Id to be indented on the left side of the page. Values are expressed in inches, centimeters, or character spaces.

Indent Right: This is used to define the Generic Id to be indented on the right side of the page. Values are expressed in inches, centimeters, or character spaces.

Indent 1st Line: This is used to define the first line of the Generic Id to be indented on the left side of the page. Values are expressed in inches, centimeters, or character spaces.

Vertical Space Before: This is used to define whether or not the Generic ID is processed with vertical space before it is placed on the defined page. Values are expressed in inches, centimeters, or character spaces.

Vertical Space After: This is used to define whether or not the Generic ID is processed with vertical space after it is placed on the defined page. Values are expressed in inches, centimeters, or character spaces.

Horizontal Space Before: This is used to define whether or not the Generic ID is processed with horizontal space before it is placed on the defined page. Values are expressed in inches, centimeters, or character spaces.

Horizontal Space After: This is used to define whether or not the Generic ID is processed with horizontal space before it is placed on the defined page. Values are expressed in inches, centimeters, or character spaces.

Page Eject Before: This is used to define whether or not the Generic ID has a page ejected before it is presented on the page. Values are expressed in integer number of pages or blank.

Page Eject After: This is used to define whether or not the Generic ID has a page ejected after it is presented on the page. Values are expressed in integer number of pages or blank.

Font Size: This is used to define the Generic ID with a specific font size other than the one specified for the entire document.

Leading: This is used to define the Generic ID with any special spacing requirements between characters other than those specified for the entire document.

Line Spacing: This is used to define the Generic ID with special line space requirements other than those specified for the entire document. Value is expressed in number of lines.

Font Name: This is used to define a special font type or style for the Generic ID other than that specified for the entire document. The <HELP> key will list available fonts for your use.

After all of the selected attributes for the selected Generic ID have been changed, press <ENTER> to save the Document Profile and return to the Generic Identifier Screen. Another Generic Identifier can be selected from this screen for attribute definition by placing the cursor on the selected Generic ID and pressing <ENTER>. However, to quit this form you must press the IISS QUIT key <PF4>. The <PF4> key will act as a quit key in all of these forms and will take you back to the previous form.

5.2.1.1.3 Headers/Footers

This is the third function of the Layout Editor. Place the cursor on the field and press <ENTER>. The Header/Footer Form will appear on your screen.

IISS Electronic Documentation System (Layout Editor) 12/ 4/87 15:09:02			
Headers/Footers			
(Function)	Define	Delete	
(Page Side)	Even	Odd	Both
(Position)	Top	Bottom	
List of Headers/Footers			
Page Side	Header/Footer		
BOTH	FOOTER		
MSC [] application			

Figure 5-6 Header/Footer Form

The input fields on this form are similar to some of the fields you have seen on the other Layout Editor screen forms. The initial input fields are:

Function: Define or Delete of header or footer

Page Side: Even or Odd or Both of the pages

Position: Top (header) or Bottom (footer) of the page

After you have selected the Function, the Page Side, and the Position; press the <ENTER> key. This will complete the form for the selected Function and Position. A Define Function will display the rest of the form. For example, if the Define Function and the Top Position was selected, the following form would appear.

IISS Electronic Documentation System (Layout Editor) 12/ 4/87 '5:13:19

Headers/Footers

(Function) (Page Side) (Position)	Define Even Top	Delete Odd Bottom	Both
from Top	0.00	IN	Start (at) TAG < >
from Bottom			Reset Page (at) TAG < >
Left Margin			
Right Margin			
Style	Center Fontsize Leading Fontname Right Justify Underline		
Text	Line (1) Line (2) Line (3) EDS Demonstration Arizona State University December 7, 1987		

HSG: 0 application

Figure 5-7 Header/Footer Form - Define

Input fields and their meanings follow:

- from Top: The value in inches, centimeters, or character spaces that the header is from the top of the page.
- from Bottom: The value in inches, centimeters, or character spaces that the footer is from the bottom of the page.
- Left Margin: The value in inches, centimeters, or character spaces the header or footer is from the left side of the page.
- Right Margin: The value in inches, centimeters, or character spaces the header or footer is from the right side of the page.
- Start (at) Tag: The tag name where the header or footer is to start.
- Reset Page (at) Tag: The tag name where the page number counter would be reset to one.

Style Center: Whether or not the header or footer is centered on the page. Value is 'x' for yes or blank for no.

Style Font Size: The size of the font for the header or footer is specified here and would be different than that specified for the entire document.

Right Justify: This defines the header or footer to be right justified. Value is 'x' for yes or blank for no.

Underline: This defines the header or footer to be underlined. Value is 'x' for yes or blank for no.

Style Leading: The definition of spacing requirements for the characters printed. Expressed in an integer number for point size.

Style Font Name: The name of the font for the header or footer if different from that of the entire document would go here.

Text Line (1): This is a line for text input of the header or footer.

Text Line (2): This is a line for text input of the header or footer.

Text Line (3): This is a line for text input of the header or footer

Text Line Macros:

- @page - to invoke header/footer on a specific page
- @time - to print the computer system time in the header/footer
- @date - to print the computer system date in the header/footer
- @counter <tag> - to count and print the number of occurrences of the specified tag

The input fields are tabbed to and after all selected fields have been changed, the Document Profile is saved by pressing the <ENTER> key. The screen form will be redisplayed with the new input fields. The IISS QUIT key <PF4> will take you back to the previous screen form. That form should be the Layout Editor Function Screen.

The input fields for the footer definition are virtually the same as that of the header definition. The delete of a header or a footer is accomplished in the same way as that of a

define function.

5.2.2 Edit Document Profile

To edit a Document Profile for a DTD, the E (dit) action code must be entered in the Action Field on this form. This Action Field is located to the left of the DTD name with the selected Document Profile. The Description Field is not available for your use. It should be noted, that Document Profile creation is the only time that the Description field can be entered.

The edit of a Document Profile is similar to the creation of a Document Profile in that the same forms are used. The E (dit) action on an existing Document Profile displays the previously defined information. You can then perform the necessary edits to that information in the Layout Editor forms. Place the cursor on the Action Field where the DTD Unit Test Plan is listed and enter an 'E'.

Now press the <ENTER> key for the system to process your E (dit) action on the selected DTD Document Profile.

1155 Electronic Documentation System (EDS) 11/ 1/87 15:32:24

Layout Editor

Action	DTD Name	Document Profile	Description
	EDSUTP	EDSLE1	EDS LAYOUT EDITOR UTP

MSG: C application

Figure 5-8 Edit Document Profile

The system will now proceed to the Layout Editor functions for the selected Document Profile. Reference Section 5.2.2 for discussion and explanation of the Layout Editor Function Screen and all subordinate screen forms and functions.

5.2.3 Copy Document Profile

The C (opy) action of the Layout Editor Screen is dependent on the forms for that one screen alone. You will want to use the C (opy) action to reduce Document Profile edit time. Copying a Document Profile and then editing that profile will be a more efficient use of the system than creating new Document Profiles. This is especially true of the highly structured documentation standards that exist on many projects and in the industry. A copy of an existing Document Profile is accomplished with these steps. Place the cursor on the Action Field on the same line as the DTD and Document Profile that you wish to copy. Enter a 'C' in that Action Field and press the <ENTER> key, see the next figure.

IISS Electronic Documentation System (EDS)			11/ 6/87 15:35:46
Layout Editor			
Action	DTD Name	Document Profile	Description
	EDSUTP	EDSLE1	EDS LAYOUT EDITOR UTP
Source DP	Destination DP	Description	
EDSLE1	EDSLE2	COPY TEST (D)	
MSG: C		application	

Figure 5-9 Layout Editor Screen - Copy

An additional form will appear on the Editor Layout Screen. This form will have three input fields on it. These input fields are Source Document Profile (DP), Destination DP, and Description of Destination DP. You must tab the cursor over to these fields and input the data. The Source DP is the Document Profile you are making a copy of. The Destination DP is the new file that you are creating in this C (copy) action. The Description field is the information field for the Destination DP. After you have entered this information into the input fields, press <ENTER> to execute the action.

The Layout Editor Screen will redisplay with the new Document Profile in the displayed list of Document Profiles. The <PF4> will cancel or quit this screen at any time.

IISS Electronic Documentation System (EDS) 11/ 6/87 15:36:26

Layout Editor

Action	DTD Name	Document Profile	Description
	EDSUTP	EDSLE2	COPY TEST (D)
	EDSUTP	EDSLE1	EDS LAYOUT EDITOR UTP

MSG: C application

Figure 5-10 Editor Layout Screen - After Copy

5.2.4 Delete Document Profile

The D (elete) action on the Layout Editor Screen is performed in a similar fashion to the C (opy) action. The D (elete) action of the Layout Editor Screen is dependent on the forms for that one screen alone. You will want to use the D (elete) action when a document profile is no longer current. A delete of an existing Document Profile is accomplished with these steps. Place the cursor on the Action Field on the same line as the DTD and Document Profile that you wish to delete. Enter a 'D' in the Action Field and press the <ENTER> key, see the next figure.

IISS Electronic Documentation System (EDS) 11/ 6/87 15:36:26

Layout Editor

Action	DTD Name	Document Profile	Description
1			
5	EDSUTP	EDSLE2	COPY TEST (D)
	EDSUTP	EDSLE1	EDS LAYOUT EDITOR UTP

OK to delete Document Profile ? EDSLE2 YES NO

MSG: C application

Figure 5-11 Layout Editor Screen - Delete

An additional form will appear on the Editor Layout Screen. This form will have verify the intent to delete. A question will appear:

OK to delete Document Profile ? DPname YES NO

The system has a default assignment for the question's answer. That default is the 'YES'. A press of the <ENTER> key would complete the D (elete) action. The Layout Editor Screen would redisplay and the DP that was deleted would no longer be on the displayed list.

```
IISS Electronic Documentation System (EDS) 11/ 1/87 15:25:20

Layout Editor

Action DTD Name Document Profile Description
1 EDSUTP EDS121 EDS LAYOUT EDITOR UTP

MSG: C application
```

Figure 5-12 Layout Editor Screen - After Delete

However, if you do not want to delete the DP, you must tab to the second response. By placing the cursor on "NO" and pressing the <ENTER> key, you would stop the D (elete) action. Again, a <PF4> key would quit the screen form and return you to a previous screen.

SECTION 6

SGML PARSER

6.1 Parsing for SGML

The SGML Parser is the component of the EDS that validates the logical structure of a document against a SGML Document Type Definition. The SGML Parser was first implemented under the MS-DOS environment by the National Bureau of Standards (NBS). The Parser was ported or migrated from the MS-DOS environment to the IISS. Portions of the code were changed to enhance its functionality in the EDS.

The validation process of the Parser ensures that the descriptive markup used within the document to delimit, or tag, the logical structures conforms to the set of rules defined by the DTD. These rules determine what tags are valid and how many times a SGML tag occurs within a document. Additionally, the Parser converts the document to its fully "marked up" state by processing all Entity references and expanding all minimized tags found within the document.

6.2 Running the Parser

The execution of the SGML Parser is accomplished by entering a set of commands on the command line. These commands involve invoking the Parser, naming the source document text file, assigning a file name for the SGML marked up file, and pressing the <return> key. The command structure is:

```
sgml -ofilename1.ext edsdtlib:dtdname.dtd filename2.ext  
<return>
```

sgml -o	Invokes the Parser
filename1.ext	This is the output file name of the marked up file with extension name after parsing the document for SGML tags.
edsdtlib:	This is the EDS logical name for the directory where the DTDs are located.
dtdname.dtd:	This is the DTD that is defined in the above directory with a .dtd extension type.
filename2.ext	This is the name of the document text file with extension name, i.e. edsdemo.doc
<return>	This is the final input for the processing of a command to the computer system.

The Parser will not resolve the inclusion of graphics files. However, a fully marked up SGML document will be the Parser's output (filename1.ext from above).

Any errors detected during the parsing of the document will be noted by the Parser. These errors will cause the Parser to return error messages to the screen display showing the errors that occurred and an approximation of where the errors occurred. The marked up file will not exist if errors were detected during parsing. These errors must be fixed in order to parse the document text file. After the errors have been corrected, the Parser must be invoked again. Only a cleanly parsed document can continue to the formatting stage of the EDS.

SECTION 7

TAGGER

7.1 Tagging a File

The tagging or "marking up" of a text file for processing into a document is done in the specific text processing system that is available. EDS allows for tagging a document in the text when creating or editing an EDS document. However, there are many documents that are not EDS documents that may be converted to EDS. This conversion is accomplished with the Tagger.

The SGML Tagger enables you to specify a translation between a document in word processing format to an SGML tagged document. A translation program must be built for each unique word processor and document type definition that will be used to process documents. A forms-driven interface is provided to assist in the pattern matching type statements that define the mapping between word processor formatting codes and document content to SGML tags. These statements are used as input for a generated LEX program that performs the actual translation of the word processing document.

7.2 Document Markup

Markup is a set of text-like commands that is added to the data of a document to imply information about the data of the document. Each significant element of the document is located and marked, thus the term markup, with a mnemonic name. This mnemonic name, the generic identifier (GI), is then associated with that document element. The definition of a document element and its mnemonic is accomplished in the DTDBLD component of the EDS. This Document Type Definition (DTD) is used to ensure the source document conforms to the logical structure defined for it.

The GI has a specific syntax associated with it. The start or beginning tag of a GI is delimited by the less-than symbol (<) and to complete the tag the delimiter greater-than symbol (>). The ending delimiter to a tag, either the start or end tag, separates the GI from the actual text of the document.

With the use of a start tag for the mnemonic there may also be an end tag. The beginning of the end tag is delimited by the less-than symbol followed by the solidus symbol (</).

Document Element = Generic Identifier (GI, mnemonic name)

GI includes:

<start tag>

</end tag>

Figure 7-1 Generic Identifier in Markup

The GI or document element is only identified in the above structure.

The physical presentation characteristics of the GI is generated separately. Establishing the relationship between the document element or GI (logical structure) and the physical characteristics for that GI (layout style) are accomplished in the Layout Editor component of the EDS.

The processing of the source document text with the GIs and their layout styles is verified with another EDS component. That component is the SGML Parser. The Parser validates the source document with its defined logical structure without processing the source document. The Parser reads the GIs, recognizes the GIs, and inserts those tags associated with the GI into the parsed file.

The Formatter is then run to complete the processing of the document. The parsed document file is mapped to the physical or layout characteristics to create the final form of the document which is generated in Postscript.

7.2.1 Manual Document Markup

The manually marking up of a source document is accomplished by you or by the authors of the source content. This writing of source content for the document can be done using any text editor available to you. EDS does not rely on any particular text editor. Any word processing file can be converted for use by EDS, see Section 7.3.

The marking up of a source document follows the basic concepts of document elements. We have defined a document structure in Section 4.0. Those identified elements have been placed within a logical structure of the document type. Those elements have been given names (GI) for use as reference back to the individual document elements. It is these mnemonic names that are used in the marking up of a source document.

You have entered these names in the DTDBLD; have defined the format parameters for them in the Layout Editor. Many of these SGML tags are apparent to you. For example:

for coverpage the SGML tag is <covpg>
ending tag is <\covpg>

for a paragraph the tag is <p0>
ending tag is <\p0>

for a major section tag the mark up would be

```
<sect>  
<sectno>SECTION 7<\sectno>  
<sectnm>TAGGER<\sectnm>
```

for subsequent sections to the major section

```
<chap>  
<chapno>7.1<\chapno>  
<chapnm>Tagging a File<\chapnm>
```

The actual document markup that you would do is predicated on your defined document structure. For further reference on the marking up of a document, see ISO 8879 and Appendices A through C of this document.

7.3 Automated Document Markup

Typically, word processing files contain both logical and layout structure information. Formatting codes (character sequences) are used to represent logical (example, paragraph) and layout (example, page break) information within the document. In order to tag a document, certain formatting codes are used to generate SGML tags when they are encountered. For example, if all section titles in a document were always centered and underlined, then a specification statement used by the Tagger would be feasible. This statement would be processed when the character sequence for center and underline is recognized. The statement would write a tag to the former word processing file.

Along with the formatting codes, the actual content of a document can be used to generate SGML tags. For example, a user can specify that the character sequence section number is recognized. The section number sequence will cause a section number to be written to the file. Specifications using a combination of content and formatting codes are also possible.

In addition to the forms-driven interface, the Tagger consists of a lexical writer, a run-time processor, and a specification storage program. Once all specification statements are entered and validated, the Tagger begins the lexical writer, Autotag, to produce a valid LEX program. This LEX program corresponds to the specification of the components supplied by the user. These components are the formatting commands of the unique word processor. The run-time processor runs the LEX to produce a C program from the specification file, compiles the result, and links it with the main program of the Tagger. The main program is used to tag a word processing document.

After the document has been tagged, it is sent to the Parser. The output of the Parser, the error report, provides an indication as to how well the document has been tagged. The Parser inserts end tags in the document by using the DTD information to decide when the tag is ended. Any necessary corrections to this newly tagged and parsed document can then be made.

Use of the Tagger and its components will not ensure that the document will be completely and correctly tagged for SGML. The efficiency of the translation from word processor to SGML is dependent on the number of defined patterns that can be interpreted to generate SGML tags. Forethought should be used in the development of these tools for this conversion process.

7.4 The Tagger

The Tagger is used in the IISS for the initial definition of a specification type. This specification is defined in the Autotag program of the Tagger. To use the run-time processor and the Tagger, it is not necessary to be in the IISS.

7.4.1 Autotag

This EDS utility will create and generate a tagger specification for SGML. The user must be able to understand the command structure of the word processing system that the specification will be built for. You must log into the IISS with a valid user id and account. The IISS Function Screen will be the next screen to appear. Place the cursor on the Function field, type in 'Autotag', and press the <ENTER> key.

The next screen form will be that of the Autotag utility. The functions available for Autotag are listed on the screen as illustrated in the next figure.

```
IISS SGML Auto Tagging Facility

Auto Tag Definition File: 

<enter> - Load Auto Tag Definition File
<pf5>   - State Definiton
<pf6>   - Character Class Definiton
<pf7>   - Pattern/Action/State Definition
<pf8>   - C Declarations
<pf16>  - Save Auto Tag Definition File

MSG: 0 application
```

Figure 7-2 Autotag - IISS SGML Auto Tagging Facility

The name of the specification file that you will define or edit must first be typed in. This name will be typed into the field for the Auto Tag Definition File. After entering the file name and pressing the <PF16> key (LOAD), the file you entered will be loaded for use in the Tagger specification forms. The <ENTER> key will load the specification file into the system for you if that file exists. If a new specification is being started the template file for the appropriate word processor must be loaded. When <ENTER> is pressed the file is saved. The functions available to you are:

PF5 State Definition
PF6 Character Class Definition
PF7 Pattern/Action/State Definition
PF8 C Declarations
PF16 Save Auto Tag Definition File

Each function has a Program Function (PF) Key defined to it. By pressing the PF Key for the desired specification (or function) you will only have to press a single key.

Each of the specification screens operate in a consistent manner. Each screen has a field at the top for entering new data, a field at the bottom of each group to indicate the end of the group, and a repeating field in the middle to display previously entered data.

Placing the cursor on a field and pressing an action key, <PF> key, allows you to modify the field by invoking a more detailed screen for the selected action. The content of the bottom field is not part of the specification and cannot be edited. Pressing <ENTER> adds the contents of the new data field to the group. Fields can be deleted by moving the cursor to the desired field and pressing <DELETE> (PF??).

Syntax checking is provided for each specification screen in order to shorten the amount of time needed to create a valid LEX program.

7.4.1.1 State Definition

The State Definition function is used to list and define states which are used in pattern/state and action definitions. States are used to signify that:

- 1.) a certain event has occurred
- 2.) specifications can be created which allow only certain actions to occur in certain states

For example, when a text SECTION <number> is recognized, the state SECTION will be set to an "on". A specification could then be defined which would recognize Section Titles (defined by <center><underline> formatting codes) only when the state SECTION was set on.

States are identifiers and may appear in any order. If a state is used in the pattern/state and action definitions without being defined in the State Definition Form, a warning is issued. A comment field is provided for each state entered in the State Definition form.

YISS SGML Auto Tagging Facility
State Definition

<enter> - Select

State

MSG: 0

application

Figure 7-3 State Definition Screen - Select

Select <ENTER>

This action will pass the user into the a second form.

YISS SGML Auto Tagging Facility
State Definition

<enter> - Insert/Update
<pf12> - Delete

State

MSG: 0

application

Figure 7-4 State Definition Screen - Other Options

Insert/Update <ENTER> This action will insert new data or update existing data in the state definition.

Delete <PF12> This action will delete the state definition listed.

7.4.1.2 Character Class Definition

The Character Class Definition form is used to list and define character class names and their regular expression definitions. The character class name serves as a mnemonic for the word processing formatting code. These class names are

used in the pattern definitions in place of a regular expression. All character class names must be defined in this form before they are used in a pattern. If the character class is not defined in this form, a warning will be issued.

IISS SGML Auto Tagging Facility Character Class Definition	
<enter> - Select	
Character Class	Regular Expression Definition
bold_on	"{#"
bold_off	"{\\""
und_on	"{X"
und_off	"{\$"
sup_on	"{)"
sup_off	"{("
sub_on	"{+"
sub_off	"{*"
aux_on	"{'"
aux_off	"{&"
dx_tilde	" >"
dx_l_brace	" ;"
dx_r_brace	" =
dx_bar	" <"
MSG: ☐ 0	application

Figure 7-5 Character Class Definition - Select

IISS SCHL Auto Tagging Facility	
Character Class Definition	
<enter>	- Insert/Update
<pf12>	- Delete
Character Class	
Regular Expression Definition	
MSG:	application

Figure 7-6 Character Class Definition - Other Options

7.4.1.3 Pattern/Action/State Definition

The Pattern/Action/State Definition function is used to list patterns and actions. The syntax of patterns is provided in the next section. An action is any valid C statement that normally occurs in the body of a procedure. The special C action "ECHO;" causes the contents of the input buffer to be

output without formatting information. The special C action "BEGIN state_name;" sets the current state to the specified state name. The current state name is reset if the state name is "0".

IISS SGML Auto Tagging Facility	
Pattern/Action/State Definition	
<enter> - Select	
Pattern	Action
.	/* anything else */
MSG: 0	
application	

Figure 7-7 Pattern/Action/State Definition - Select

YISS SGML Auto Tagging Facility Pattern/Action/State Definition	
<enter> - Insert/Update/Select (Action or State) <pf12> - Delete	
Pattern	
Action	
State	
MSG: ☐ 0	application

Figure 7-8 Pattern/Action/State Definition - Other Options

YISS SGML Auto Tagging Facility C Code	
<enter> - Insert/Update <pf12> - Delete	
Code	
MSG: ☐ 1 entered	application

Figure 7-9 C Code

7.4.1.4 C Declarations

This C definition form is used to insert C statements for the LEX module yylex, this module name is a unique name to the Lexical Writer and should not be changed. C statements are used to define variables referenced in actions and to perform initialization of the output document file. For example, you might wish to output several tags before beginning the Tagger processing of your word processing file.

IISS SGML Auto Tagging Facility C Declarations Section	
<enter> - Select	
Code	
MSG: ☐ 0	application

Figure 7-10 C Declarations Section

7.4.1.4.1 Pattern Specification Syntax

Patterns represent elements of a document that are to be tagged. These patterns consist of strings and operators. A string is a sequence of printable characters. An operator is one of the following characters:

" \ [] ^ - ? . * + | () \$ / { } % <>

The characters <space>, <> (angle brackets), ^ (karat), % (percent), and \$ (dollar sign) are special reserved characters to LEX and should be enclosed in double quotes or preceded by a backslash. Warnings will be issued for improper use of these characters.

7.4.1.5 Save Auto Tag Definition File

The Save Auto Tag Definition File function is used to save the defined Autotag data. The selection of this function will return you to the previous screen form.

7.4.2 The Lexical Writer

The Lexical Writer of the EDS uses the specification defined in the Autotag utility. The Lexical Writer is not accessed in the IISS. This component is invoked on the command line, i.e. a '\$' prompt:

```
LEX specificationname.L <RETURN>
```

This component of the EDS Tagger will create a C program, LEXxx.C. The C program name is specified by the system; the name LEXxx.C is the actual name assigned that C program.

7.4.3 Run-Time Processor

You now have a C program from the Lexical Writer. This program must now be compiled and linked. The command:

```
CC LEXxx <RETURN>
```

will compile the application and create an object file, LEXxx.OBJ.

The application must now be linked. The command:

```
LINK LEXxx,[dxtag]dstag/i=(DX,DXPRE) <RETURN>
```

will link the application. The directory [dxtag] and library dxtag contain the runtime support modules for DECDX document taggers.

The application is ready to run to tag a document in SGML. This application will take the word processor file name that you supply and tag it. The C program LEXxx will write the newly tagged file into the file name that you specify. The

following is the command to enter to begin the application.

RUN LEXxx <RETURN>

Two input fields will appear on the screen. They are:

file?
out?

you must enter the word processor filename in the file input field. The input for the out field is a filename for the SGML tagged document, so whatever name you input here will become the newly tagged and parsed SGML document file name.

The application is built for the specification defined in Autotag. The Run-time application tags the document and parses it also. The former word processor document is now an SGML tagged document for your use.

SECTION 8

GRAPHICS SUPPORT

8.1 EDS and Graphics

EDS is used to combine text and graphics into a single, cohesive, integrated document. This eliminates the need for manually "cutting and pasting" the graphics into the text. There are several externally generated graphics that can be included in an EDS document. You can use IISS screen dumps, you can use Apple MacIntosh MacPaint files, you can use SDRC I-DEAS files, and other formatted (human readable) Postscript files for inclusion in an EDS document.

8.2 IISS Screen Dumps

The EDS was developed to assist the user in conforming to documentation standards and to integrate text and graphics. EDS is automatically capable of using certain keyboard character sequences to capture IISS screen dumps. There is a character sequence which can be used for capturing these IISS screens, <CTRL p>.

The <CTRL p> character sequence captures IISS screen dumps in either the Postscript format or the ASCII file format. The EDS has certain links defined for it. The link for your account and/or your application may be defined for the Postscript format or the ASCII format. You must be correctly linked to the PSPRINT routine from DEVDRV in order to produce the Postscript pictures from the IISS screens. Please note that you may be able to save these IISS screens but, in order to print them there must be a printer that supports Postscript available to you.

8.3 Apple MacIntosh MacPaint Files

The generation of Apple MacPaint graphics for the general computer user has been widespread in the industry. However, as with most of these independent systems there has been no easy way to include these graphics into documents. The EDS was developed to include externally generated graphics into the EDS document. The MacIntosh MacPaint files are converted to a language that the EDS can process, this language is Postscript.

The content of and the format of the converted file is based on the EPSF structure conventions documented by Adobe Systems, Incorporated. This convention is available from Adobe Systems, Incorporated and Structural Dynamics Research Corporation.

An EDS application program, the MPEPSF translator, is available for the Apple MacIntosh that translates and converts MacPaint files to an encapsulated Postscript file (EPSF). This application must be executed on the Apple MacIntosh. The translation of the MacPaint files is not dependent on the IISS nor on any additional EDS components.

This EPSF file is an ASCII text file containing Postscript language statements that describe the MacPaint file. EPSF files can be transferred to the VAX via a communication package and then used in an EDS document.

8.3.1 MacPaint File

The icon driven software available on the Apple MacIntosh is used by the user to create and save a MacPaint file. The conversion program is selected from the MacIntosh menu.

8.3.2 MPEPSF

Now that you are in the MPEPSF application, you can convert a MacPaint file. The MacPaint document can be viewed by selecting the OPEN menu item from the FILE function. The name of the MacPaint file must be selected for any processing to continue.

Conversion of the MacPaint file is begun by selecting the CONVERT item from the FILE function. An output file name must be supplied by you before the file is converted. After the file is converted, exit from the MPEPSF application by selecting the QUIT item from the FILE menu.

8.3.3 File Transfer

The converted file must be sent to the VAX. Any file transfer program that is available to you should work for this file transfer, i.e. Kermit, Apple MacTerminal, Red Ryder. The converted and transferred file is ready for printing to any Postscript compatible print device.

8.3.4 Print

The use of these files in an EDS document requires an include statement in the document source. The syntax for this is contained in the figure markup statements for the figure:

```
<figure>
<graphic file = "filename.epsf" place="fixed"
.
</graphic>
.
</figure>
```

See Appendix C for an example of a marked up EDS source document.

These files require the following print qualifiers for the VAX when printing as an individual graphic and not in an EDS document:

```
/NOHEAD
/NOTRAIL
/NOFLAG
/NOBURST
```

SECTION 9

EDS FORMATTER

9.1 Formatting the EDS Document

The EDS Formatter reads an SGML tagged document, one successfully run through the SGML Parser, and uses the formatting information contained in the Document Profile to determine the layout or format style of the document. Formatter output is in the form of Postscript language statements. These statements, when recognized by a Postscript output device, will format the document as defined in the DTD, the Document Profile, and the SGML Parser. These Postscript commands are printed and, also, stored in a file. The EDS Formatter also inserts the graphics into the formatted text.

9.2 EDSFMT

This is the program module of the EDS that formats the document. The user must log into the IISS with a valid user id and account. The IISS Function Screen will appear on the screen after successful logon. Place the cursor on the Function field, enter EDSFMT, and press <ENTER>. The EDS Document Formatter Screen will appear next. The Document Formatter Screen is forms based and has input fields for the input filename, the Document Profile, and the output filename.

IISS Electronic Documentation System (EDS) 12, 4/87 15:34:20

Document Formatter

Document Name: didbld.cir

Document Profile: didbld

Output File: didbld.out

MSO application

Figure 9-1 Document Formatter Screen

The input fields on the screen are Document Name, Document Profile, and Output File.

Document Name: This is the name of the Parsed file, reference Section 6, not the the name of the document text file.

Document Profile: This is the name of the Document Profile to use for the formatting requirements on this document.

Output File: This is the name of the generated output file from the formatter process, i.e. filename.out.

These input fields must be entered in order to successfully format a document. An incorrect filename or Document Profile name will return an error message to the screen. You will then need to re-enter the necessary input. The <ENTER> key will complete the form and begin the format process.

After the document has been formatted another form will appear. This form will ask you if you want to print the output. The default assignment for this message is 'YES', a press of the <ENTER> key will send the formatted document to a print device. A 'NO' response will bring you back to the previous form and the QUIT key <PF4> will also bring you back to the previous form, the Document Formatter Screen.

You have now successfully formatted a document. Your document includes externally generated graphics.

APPENDIX A

EDS DTDBLD Document Type Definition

The following pages are computer listings of a document type definition.

```

<!--
  DEMO.DTD - Document Type Definition for Demo Documents

  This Document Type Definition should be used for all ICAM
  documents written by IISS coalition members. Please do not
  make any changes to this document.

  Created : 15-November-1987 Steve Barker
  Revised : 15-November-1987 Steve Barker

-->

<!DOCTYPE demodoc [

  <!-- Define a list of entity refs that can be used for long string names -->

  <ENTITY ADL          "Application Definition Language" >
  <ENTITY AI           "Application Interface" >
  <ENTITY AP           "Application Process" >
  <ENTITY CDC          "Control Data Corporation" >
  <ENTITY CM           "Configuration Management" >
  <ENTITY CDM          "Common Data Model" >
  <ENTITY DM           "Documentation Management" >
  <ENTITY DS           "Development Specification" >
  <ENTITY EDS          "Electronic Documentation System" >
  <ENTITY FD           "Form Definition" >
  <ENTITY FDFE         "Forms Driven Form Editor" >
  <ENTITY FDL          "Forms Definition Language" >
  <ENTITY FE           "Form Editor" >
  <ENTITY FLAN         "Forms Language Compiler" >
  <ENTITY FP           "Form Processor" >
  <ENTITY IISS         "Integrated Information Support System" >
  <ENTITY ISO          "International Standards Organization" >
  <ENTITY LOS          "Layout Optimization System" >
  <ENTITY NDDL         "Neutral Data Definition Language" >
  <ENTITY NDML         "Neutral Data Manipulation Language" >
  <ENTITY NTM          "Network Transaction Manager" >
  <ENTITY OS           "Operating System" >
  <ENTITY PS           "Product Specification" >
  <ENTITY SDRG         "Structural Dynamics Research Corporation" >
  <ENTITY SGML         "Standard Generalized Markup Language" >
  <ENTITY TOC          "Table of Contents" >
  <ENTITY UI           "User Interface" >
  <ENTITY UIMS         "User Interface Management System" >
  <ENTITY UIS          "User Interface Services" >
  <ENTITY UTP          "Unit Test Plan" >
  <ENTITY VT           "Virtual Terminal" >

  <!-- Define Document Markup for a DEMO Document -->

  <ELEMENT demodoc      o o (body) >
  <ELEMENT body         - - (h1, {para0 | para1 | para2 | para3 | para4 | figure
  <ELEMENT h1           - o (#PCDATA) >
  <ELEMENT para0        - o (#PCDATA) >
  <ELEMENT para1        - o (#PCDATA) >
  <ELEMENT para2        - o (#PCDATA) >
  <ELEMENT para3        - o (#PCDATA) >
  <ELEMENT para4        - o (#PCDATA) >
  <ELEMENT figure       - - ((figbody | graphic | report) , figdesc?) >
  <ELEMENT figdesc      - - (figref?, fignum?, figtitle) >
  <ELEMENT figref       - o (#PCDATA) >
  <ELEMENT fignum       - o (#PCDATA) >
  <ELEMENT figtitle     - o (#PCDATA) >
  <ELEMENT list         - - (item)+ >

```

```

<!ELEMENT item      - O  ($PCDATA) >
<!ELEMENT report    - -  ($PCDATA) >
<!ELEMENT graphic   - -  ($PCDATA) >
<!ELEMENT figbody   - -  ($PCDATA) >

<!-- Define Attributes -->

<!-- ATTLIST graphic
file      CDATA      #REQUIRED
fmt       (epsf | ipic) "epsf"
place     (fixed | float) "float"
posx      (left | center | right) "center"
posy      (top | middle | bottom | any) "middle"
width     NMTOKENS      "6.5IN"
depth     NMTOKENS      "5.5IN"
border    NMTOKENS      "NONE"
shadow    NMTOKENS      "NONE"
>
<!-- ATTLIST report
file      CDATA      #REQUIRED
fmt       (text)      "text" >
]>

```

APPENDIX B

EDS Unit Test Plan Parsed Document

The following pages are computer listings of an EDS unit test plan manual that has been parsed.

```
[ICAMUTP]
[FRONT]
[COVPG]
[SYSTTL]
[TTL]INTEGRATED INFORMATION SUPPORT SYSTEM (IISS)
[/TTL]
[/SYSTTL]
[DOCTTL]
[TTL]USER INTERFACE/VIRTUAL TERMINAL INTERFACE (UI/VTI)
[/TTL]
[TTL]COMPUTER PROGRAM UNIT TEST PLAN
[/TTL]
[TTL]CONFIGURATION ITEM: EDS SGML Parser
[/TTL]
[/DOCTTL]
[DOCDATE]31 March 1988
[/DOCDATE]
[RECBK]
[RECTTL]PREPARED FOR:
[/RECTTL]
[ALINE]SYSTRAN CORPORATION
[/ALINE]
[ALINE]4126 LINDEN AVE.
[/ALINE]
[ALINE]DAYTON, OHIO 45432-3068
[/ALINE]
[/RECBK]
[SNDBLK]
[SNDTTL]PREPARED BY:
[/SNDTTL]
[ALINE]STRUCTURAL DYNAMICS RESEARCH CORPORATION
[/ALINE]
[ALINE]2000 EASTMAN DRIVE
[/ALINE]
[ALINE]MILFORD, OHIO 45150
[/ALINE]
[/SNDBLK]
[/COVPG]
[CONTENTS]
[TTL]TABLE OF CONTENTS
[/TTL]
[TTL2]Page
[/TTL2]
[XSECTNO]SECTION 1.0
[/XSECTNO]
[XSECTNM]GENERAL .....
[/XSECTNM]
[PGNO]1-1
[/PGNO]
[XCHAPNO]1.1
[/XCHAPNO]
[XCHAPNM] Purpose .....
[/XCHAPNM]
[PGNO]1-1
[/PGNO]
[XCHAPNO]1.2
[/XCHAPNO]
[XCHAPNM] Project References .....
[/XCHAPNM]
[PGNO]1-1
[/PGNO]
[XCHAPNO]1.3
[/XCHAPNO]
[XCHAPNM] Terms and Abbreviations .....
[/XCHAPNM]
```

```
[PGNO]1-2
[/PGNO]
[XSECTNO]SECTION 2.0
[/XSECTNO]
[XSECTNM]DEVELOPMENT ACTIVITY .....
[/XSECTNM]
[PGNO]2-1
[/PGNO]
[XCHAPNO]2.1
[/XCHAPNO]
[XCHAPNM] Statement of Pretest Activity .....
[/XCHAPNM]
[PGNO]2-1
[/PGNO]
[XCHAPNO]2.2
[/XCHAPNO]
[XCHAPNM] Pretest Activity Results .....
[/XCHAPNM]
[PGNO]2-1
[/PGNO]
[XSECTNO]SECTION 3.0
[/XSECTNO]
[XSECTNM]SYSTEM DESCRIPTION .....
[/XSECTNM]
[PGNO]3-1
[/PGNO]
[XCHAPNO]3.1
[/XCHAPNO]
[XCHAPNM] System Description .....
[/XCHAPNM]
[PGNO]3-1
[/PGNO]
[XCHAPNO]3.2
[/XCHAPNO]
[XCHAPNM] Testing Schedule .....
[/XCHAPNM]
[PGNO]3-1
[/PGNO]
[XCHAPNO]3.3
[/XCHAPNO]
[XCHAPNM] First Location Testing .....
[/XCHAPNM]
[PGNO]3-1
[/PGNO]
[XCHAPNO]3.4
[/XCHAPNO]
[XCHAPNM] Subsequent Location Testing .....
[/XCHAPNM]
[PGNO]3-2
[/PGNO]
[XSECTNO]SECTION 4.0
[/XSECTNO]
[XSECTNM]SPECIFICATIONS AND EVALUATIONS .....
[/XSECTNM]
[PGNO]4-1
[/PGNO]
[XCHAPNO]4.1
[/XCHAPNO]
[XCHAPNM] Test Specifications .....
[/XCHAPNM]
[PGNO]4-1
[/PGNO]
[XCHAPNO]4.2
[/XCHAPNO]
[XCHAPNM] Test Methods and Constraints .....
[/XCHAPNM]
```

```
[PGNO]4-1
[/PGNO]
[XCHAPNO]4.3
[/XCHAPNO]
[XCHAPNM] Test Progression .....
[/XCHAPNM]
[PGNO]4-2
[/PGNO]
[XCHAPNO]4.3
[/XCHAPNO]
[XCHAPNM] Test Evaluation .....
[/XCHAPNM]
[PGNO]4-2
[/PGNO]
[XSECTNO]SECTION 5.0
[/XSECTNO]
[XSECTNM]TEST PROCEDURES .....
[/XSECTNM]
[PGNO]5-1
[/PGNO]
[XCHAPNO]5.1
[/XCHAPNO]
[XCHAPNM] Test Description .....
[/XCHAPNM]
[PGNO]5-1
[/PGNO]
[XCHAPNO]5.2
[/XCHAPNO]
[XCHAPNM] Test Control .....
[/XCHAPNM]
[PGNO]5-1
[/PGNO]
[XCHAPNO]5.3
[/XCHAPNO]
[XCHAPNM] Test Procedures .....
[/XCHAPNM]
[PGNO]5-1
[/PGNO]
[LISTAPPX]
[TTL]Appendices
[/TTL]
[XAPPNO]A-1
[/XAPPNO]
[XAPPNM]Valid SGML Validation Suite Test Documents ..
[/XAPPNM]
[PGNO]A-1
[/PGNO]
[XAPPNO]B-1
[/XAPPNO]
[XAPPNM]Invalid SGML Validation Suite Test Documents
[/XAPPNM]
[PGNO]B-1
[/PGNO]
[/LISTAPPX]
[/CONTENTS]
[/FRONT]
[BODY]
[SECT]
[SECTNO]SECTION 1
[/SECTNO]
[SECTNM]GENERAL
[/SECTNM]
[CHAP]
[CHAPNO]1.1
[/CHAPNO]
[CHAPNM]Purpose
```

```
/CHAPNM]
[P0]This Unit Test Plan (UTP) establishes the methodology and procedures
used to adequately test the capabilities of the computer program identified
as the SGML Parser. The SGML Parser is one configuration item of the
Integrated Information Support System (IISS) Electronic Documentation System (ED
[/P0]
[/CHAP]
[CHAP]
[CHAPNO]1.2
[/CHAPNO]
[CHAPNM]Project References
[/CHAPNM]
[LIST]
[I1]\[1] Systran,
[UL]ICAM Documentation Standards
[/UL],
[UL], ISO 8879,
IDS150120000C, 5 September 1983.
[/I1]
[I1]\[2] International Organization for Standardization,
[UL]Information Processing - Text and Office Systems -
Standard Generalized Markup Language (SGML)
[/UL], ISO 8879,
15 October 1986.
[/I1]
[I1]\[3] International Organization for Standardization,
[UL]Office
Document Architecture/Office Document Interchange
Format
[/UL], ISO/DP 8613/1-6, October 1985 (Draft).
[/I1]
[I1]\[4] American National Standards Institute,
[UL]American
National Standard for Information Systems - Computer
Graphics - Metafile for the Storage and Transfer of
Picture Description Information
[/UL], ANSI X/3.122-1986,
August 27, 1986.
[/I1]
[I1]\[5] Structural Dynamics Research Corporation,
[UL]Form
Processor User's Manual
[/UL], UM 620244200A, 16 February 1987.
[/I1]
[I1]\[6] Structural Dynamics Research Corporation,
[UL]Virtual
Terminal Operator Guide
[/UL], OM 620244000A, 16 February 1987.
[/I1]
[I1]\[7] M.E. Lesk,
[UL]LEX - Lexical Analyzer Generator, IS
Workbench for VAX/VMS Programmers Guide
[/UL].
[/I1]
[I1]\[8] Structural Dynamics Research Corporation,
[UL]Form
Processor Development Specification
[/UL], DS 620244700A, 16 February 1987
[/I1]
[/LIST]
[/CHAP]
[CHAP]
[CHAPNO]1.3
[/CHAPNO]
[CHAPNM]Terms and Abbreviations
[/CHAPNM]
[LIST]
```

[I2]
[UL]American Standard Code for Information Interchange (ASCII)
[/UL]:
The character set defined by ANSI x3.4 and used by most computer vendors.
[/I2]
[I2]
[UL]Attribute
[/UL]: A characteristic used to qualify an element within a document.
[/I2]
[I2]
[UL]Character Set
[/UL]: A mapping of a character repertoire onto a code set such that each character is associated with its coded representation.
[/I2]
[I2]
[UL]Compound Document
[/UL]: A document which may contain mixed content (text, graphics, etc.).
[/I2]
[I2]
[UL]Computer Graphics Metafile (CGM)
[/UL]: A standard file format for the storage and retrieval of picture description information.
[/I2]
[I2]
[UL]Computer Program Configuration Item (CPCI)
[/UL]: An aggregation of computer programs or any of their discrete portions, which satisfies an end-use function.
[/I2]
[I2]
[UL]Conforming SGML Application
[/UL]: An SGML application that requires documents to be conforming SGML documents, and whose documentation meets the requirements of this International Standard.
[/I2]
[I2]
[UL]Context-Directed Editor
[/UL]: An EDS application which guides the user through the process of document creation and revision by using the document type definition as a model for which logical elements may be included in the document.
[/I2]
[I2]
[UL]Descriptive Markup
[/UL]: Information added to a document that enables an application program to process the document.
[/I2]
[I2]
[UL]Document Type Definition (DTD)
[/UL]: Rules determined by an application that apply SGML to the markup of documents of a particular type. A document type definition includes a formal specification, expressed in a document type declaration, of the element types, element relationships and attributes, and references that can be represented by markup. It thereby defines the vocabulary of the markup for which SGML defines the syntax. A DTD can also include comments that describe the semantics of elements and attributes, and any application conventions.
[/I2]

[I2]
[UL]Electronic Documentation System (EDS)
[/UL]: An integrated set
of software tools and application programs which operate upon a
document through various stages of a document life cycle
consisting of editing (creating/revising), formatting, imaging,
storage, and transferring.
[/I2]
[I2]
[UL]Element
[/UL]: A component of the hierarchical structure defined
by a document type definition; it is identified in a document
instance by descriptive markup, usually a start-tag and end-tag.
[/I2]
[I2]
[UL]Element Declaration
[/UL]: A markup declaration that contains
the formal specification of the part of an element type
definition that deals with the content and markup minimization.
[/I2]
[I2]
[UL]Entity
[/UL]: A collection of characters that can be referenced
as a unit.
[/I2]
[I2]
[UL]Entity Declaration
[/UL]: A markup declaration that assigns an SGML name
to an entity so that it can be referenced.
[/I2]
[I2]
[UL]Entity Reference
[/UL]: A reference that is replaced by an entity.
[/I2]
[I2]
[UL]Field
[/UL]: Two-dimensional space on a terminal screen.
[/I2]
[I2]
[UL]Form
[/UL]: A structured view which may be imposed on windows or
other forms. A form is composed of fields. These fields may be
defined as forms, items, or windows.
[/I2]
[I2]
[UL]Form Definition (FD)
[/UL]: Form definition Language after
compilation. It is read at run-time by the Form Processor.
[/I2]
[I2]
[UL]Form Definition Language (FDL)
[/UL]: The language in which
electronic forms are defined.
[/I2]
[I2]
[UL]Form Editor (FE)
[/UL]: A subset of the IIS User Interface that
is used to create definitions of forms. The FE consists of the
Forms Driven Form Editor and the Forms Language Compiler.
[/I2]
[I2]
[UL]Form Hierarchy
[/UL]: A graphic representation of the way in
which forms, items, and windows are related to their parent
form.
[/I2]

[I2]
[UL]Form Language Compiler (FLAN)
[/UL]: A subset of the FE that
consists of a batch process that accepts a series of form
definition language statements and produces form definition
files as output.
[/I2]
[I2]
[UL]Form Processor (FP)
[/UL]: A subset of the IISS User Interface
that consists of a set of callable execution-time routines
available to an application program for form processing.
[/I2]
[I2]
[UL]Forms Driven Form Editor (FDPE)
[/UL]: A subset of the FE which
consists of a forms-driven application used to create Form
Definition files interactively.
[/I2]
[I2]
[UL]Generic Identifier
[/UL]: A name that identifies the element type of an
element.
[/I2]
[I2]
[UL]GI
[/UL]: Generic Identifier.
[/I2]
[I2]
[UL]IISS Function Screen
[/UL]: The first screen that is displayed
after logon. It allows the user to specify the function to
access and the device type and device name on which to work.
[/I2]
[I2]
[UL]Integrated Information Support System (IISS)
[/UL]: A test
computing environment used to investigate, demonstrate, and test
the concepts of information management and information
integration in the context of Aerospace Manufacturing. The IISS
addresses the problems of integration of data resident on
heterogeneous data bases supported by heterogeneous computers
interconnected via a Local Area Network.
[/I2]
[I2]
[UL]Item
[/UL]: A non-decomposable area of a form in which
hard-coded descriptive text may be placed and the only defined
areas where user data may be input/output.
[/I2]
[I2]
[UL]Layout Style
[/UL]: The specification of format and presentation
for logical elements.
[/I2]
[I2]
[UL]Layout Structure
[/UL]: The hierarchy of all layout elements
(pages, frames, blocks, etc.) for a document.
[/I2]
[I2]
[UL]Logical Structure
[/UL]: The hierarchy of all logical elements
(paragraphs, sections, etc.) within a document.
[/I2]
[I2]

[UL]Markup
[/UL]: Text that is added to the data of a document in order to convey information about it.
[/I2]
[I2]
[UL]Markup Minimization
[/UL]: A feature of SGML that allows markup to be minimized by shortening or omitting tags, or shortening entity references.
[/I2]
[I2]
[UL]Message
[/UL]: Descriptive text which may be returned in the standard message line on the terminal screen. Messages are used to warn of errors or provide other user information.
[/I2]
[I2]
[UL]Message Line
[/UL]: A line on the terminal screen that is used to display messages.
[/I2]
[I2]
[UL]Operating System (OS)
[/UL]: Software supplied with a computer which allows it to supervise its own operations and manage access to hardware facilities such as memory and peripherals.
[/I2]
[I2]
[UL]Page
[/UL]: Instance of forms in windows that are created whenever a form is added to a window.
[/I2]
[I2]
[UL]Paging and Scrolling
[/UL]: A method which allows a form to contain more data than can be displayed at one time with provisions for viewing any portion of the data buffer.
[/I2]
[I2]
[UL]Parser
[/UL]: An application program that determines how closely a document conforms to a document type definition which defines a specific documentation standard.
[/I2]
[I2]
[UL]Physical Device
[/UL]: A hardware terminal.
[/I2]
[I2]
[UL]Previous Cursor Position
[/UL]: The position of the cursor when the previous edit command was issued.
[/I2]
[I2]
[UL]Qualified Name
[/UL]: The name of a form, item, or window preceded by the hierarchy path so that it is uniquely identified.
[/I2]
[I2]
[UL]Standard Generalized Markup Language (SGML)
[/UL]: A language for describing document structures, consisting of descriptive markup which is added to a document to indicate where logical elements such as sections and paragraphs begin and end.
[/I2]
[I2]

[UL]Subform
[/UL]: A form that is used within another form.
[/I2]
[I2]
[UL]Tag
[/UL]: Descriptive markup indicating the start or end of a logical element.
[/I2]
[I2]
[UL]Tagger
[/UL]: An application program which provides a mechanism for automatically tagging existing documents which have been created by word processing systems.
[/I2]
[I2]
[UL]User Interface (UI)
[/UL]: IISS subsystem that controls the user's terminal and interfaces with the rest of the system. The UI consists of two major subsystems: The User Interface Development System (UIDS) and the User Interface Management System (UIMS).
[/I2]
[I2]
[UL]User Interface Management System (UIMS)
[/UL]: The run-time UI.
It consists of the Form Processor, Virtual Terminal, Application Interface, the User Interface Services, and the Text Editor.
[/I2]
[I2]
[UL]User Interface Services (UIS)
[/UL]: A subset of the IISS User Interface that consists of a package of routines that aid users in controlling their environment. It includes message management, change password, and application definition services.
[/I2]
[I2]
[UL]User Interface/Virtual Terminal Interface (UI/VTI)
[/UL]:
Another name for the User Interface.
[/I2]
[I2]
[UL]Virtual Terminal (VT)
[/UL]: A subset of the IISS User Interface that performs the interfacing between different terminals and the UI. This is done by defining a specific set of terminal features and protocols which must be supported by the UI software which constitutes the virtual terminal definition. Specific terminals are then mapped against the virtual terminal software by specific software modules written for each type of real terminal supported.
[/I2]
[I2]
[UL]Virtual Terminal Interface (VTI)
[/UL]: The callable interface to the VT.
[/I2]
[I2]
[UL]Window
[/UL]: Dynamic area of a terminal screen on which predefined forms may be placed at run-time.
[/I2]
[I2]
[UL]Window Manager
[/UL]: A facility which allows the following to be manipulated: size and location of windows, the device on

which an application is running, the position of a form within a window. It is part of the Form Processor.

```
[/I2]
[/LIST]
[/CHAP]
[/SECT]
[SECT]
[SECTNO]SECTION 2
[/SECTNO]
[SECTNM]DEVELOPMENT ACTIVITY
[/SECTNM]
[CHAP]
[CHAPNO]2.1
[/CHAPNO]
[CHAPNM]Statement of Pretest Activity
[/CHAPNM]
[PO]During system development, the computer programs were tested progressively. Functionality was incrementally tested, and as bugs were discovered by this testing, the software was corrected.
[/PO]
[PO]The starting point for the development of the EDS SGML Parser was a Parser implementation for the MS-DOS operating system developed by the National Bureau of Standards (NBS) as part of the CALS initiative. For EDS, the Parser was ported from MS-DOS to the operating systems supported by IISS. As the code was ported, changes were made to eliminate the use of temporary files, fix minor bugs, and add additional functionality needed to support EDS.
[/PO]
[PO]The NBS also developed an SGML Parser validation suite consisting of a number of files that test specific functionality outlined in the SGML standard (ISO 8879). The validation suite will serve as the basis for the EDS SGML Parser Unit Test Plan.
[/PO]
[/CHAP]
[CHAP]
[CHAPNO]2.2
[/CHAPNO]
[CHAPNM]Pretest Activity Results
[/CHAPNM]
[PO]The EDS SGML Parser successfully passed the NBS validation suite. A few minor errors were found in the validation suite files themselves and these were corrected.
[/PO]
[/CHAP]
[/SECT]
[SECT]
[SECTNO]SECTION 3
[/SECTNO]
[SECTNM]SYSTEM DESCRIPTION
[/SECTNM]
[CHAP]
[CHAPNO]3.1
[/CHAPNO]
[CHAPNM]System Description
[/CHAPNM]
[PO]The SGML Parser is the component of the Electronic Documentation System that validates the logical structure of a document against a previously built SGML Document Type Definition. It insures that the descriptive markup in the document conforms to the set of rules defined by the DTD. The DTD determines what generic identifiers are valid at any point in the document and how many times they may occur.
[/PO]
[PO]In addition to validating the markup, the Parser also converts the document to its fully marked up state by processing all Entity references, Attributes, and expands all minimized generic identifiers found within the document.
[/PO]
[/CHAP]
```

```
[CHAP]
[CHAPNO]3.2
[/CHAPNO]
[CHAPNM]Testing Schedule
[/CHAPNM]
[PO]Since the NBS validation suite is the basis for the Parser test plan,
execution of this unit test plan is not dependent upon any other components
of either EDS or IISS.
[/PO]
[/CHAP]
[CHAP]
[CHAPNO]3.3
[/CHAPNO]
[CHAPNM]First Location Testing
[/CHAPNM]
[PO]These tests of the SGML Parser require the following:
[/PO]
[LIST]
[I]Equipment: Air Force VAX, terminals supported by the Virtual Terminal
[/I]
[I]
[/I]
[I]Support Software: C run-time libraries
[/I]
[I]
[/I]
[I]Personnel: one integrator familiar with EDS
[/I]
[I]
[/I]
[I]Training: the EDS User Manual has been previously delivered
[/I]
[I]
[/I]
[I]Deliverables: the SGML Parser CPCI
[/I]
[I]
[/I]
[I]Test Materials: All tests are run using the NBS SGML validation
suite files
[/I]
[I]
[/I]
[I]Security Considerations: None.
[/I]
[I]
[/I]
[/LIST]
[/CHAP]
[CHAP]
[CHAPNO]3.4
[/CHAPNO]
[CHAPNM]Subsequent Location Testing
[/CHAPNM]
[PO]The requirements listed above must be met. The command procedure
PARVAL.COM can be run to execute the entire NBS SGML validation suite.
The output file PARVAL.TST can then be compared using the VAX DIFF
command to the file PARVAL.SAV under IISS CM to insure that the unit
test ran correctly.
[/PO]
[/CHAP]
[/SECT]
[SECT]
[SECTNO]SECTION 4
[/SECTNO]
[SECTNM]SPECIFICATIONS AND EVALUATIONS
```

```
[/SECTNM]
[CHAP]
[CHAPNO]4.1
[/CHAPNO]
[CHAPNM]Test Specification
[/CHAPNM]
[PO]The Unit Test Plan is based on covering specific functionality of the
SGML Parser as outlined in the EDS Development Specification (DS) and the
SGML ISO 8879 reference manual.
[/PO]
[PO]The Parser is tested by using the input files of the NBS validation
suite. These files were developed to test the level of conformance
and the correctness of Parser implementations to the SGML standard.
Each validation suite test file tests one specific feature of the standard.
The first set of files, listed in Appendix A and starting with the letter
G, tests sequences of SGML language statements that are valid. No errors
should be detected by the Parser when parsing any of the valid test cases.
The second set of files, listed in Appendix B and starting with the
letter I, are test sequences of SGML language statements that are invalid
and should produce an error when they are parsed. To execute the validation
suite, each file can be parsed individually by the tester, or a command
procedure PARVAL.COM can be executed to parse all files at once.
[/PO]
[PO]The validation suite is the Unit Test for the EDS SGML Parser. The
objective of the test is to insure that the SGML Parser parses all
validation suite files in the correct manner.
[/PO]
[/CHAP]
[CHAP]
[CHAPNO]4.2
[/CHAPNO]
[CHAPNM]Testing Methods and Constraints
[/CHAPNM]
[PO]The tests outlined in Section 5 can be executed in any order. The required
input is given for each test in the form of a validation suite test file name.
A list of the test files are given
in Appendices A & B. Appendix A contains those tests that should run
without error, while Appendix B contains those tests that should produce
an error.
[/PO]
[PO]The tester can either manually parse one file at a time, or run a command
file to execute the entire validation suite. The name of this
command file is PARVAL.COM and is under IISS Configuration Management.
[/PO]
[PO]No additional constraints are placed on this unit test
besides those listed in Sections 5.2 and 3.3 of this document.
[/PO]
[/CHAP]
[CHAP]
[CHAPNO]4.3
[/CHAPNO]
[CHAPNM]Test Progression
[/CHAPNM]
[PO]The validation suite test files may be executed in any order.
[/PO]
[/CHAP]
[CHAP]
[CHAPNO]4.4
[/CHAPNO]
[CHAPNM]Test Evaluation
[/CHAPNM]
[PO]The NBS validation suite test files are divided into two categories -
test documents that should parse without error, and test documents that
are invalid and should produce an error. The files that begin with the letter
G should NOT produce any errors. The files that begin with the letter I
SHOULD produce an error message.
```

```
[/P0]
[P0]The test results are evaluated by checking the Parser output to make sure
that no error messages have been generated for G files and that error
messages are generated for all I files.
[/P0]
[P0]The SGML Parser will stop parsing the test document and generate an error
message of the form ERROR:... when an error in the document is found. The
accuracy of this Unit Test Plan is dependent upon the observation by the tester
of the output of the Parser.
[/P0]
[P0]If the tester runs PARVAL.COM then the output file PARVAL.TST is
produced. This file may be compared to the file PARVAL.SAV under IISS
Configuration Management using the VAX DIFF command. For the test
to be successful the files should match exactly.
[/P0]
[/CHAP]
[/SECT]
[SECT]
[SECTNO]SECTION 5
[/SECTNO]
[SECTNM]TEST PROCEDURES
[/SECTNM]
[CHAP]
[CHAPNO]5.1
[/CHAPNO]
[CHAPNM]Test Description
[/CHAPNM]
[P0]The Unit Test plan is executed by parsing all test documents listed in
Appendices A and B and observing the output of the Parser for each test.
As mentioned above, the test documents listed in Appendix A are valid
SGML documents and should parse without error. Those listed in Appendix
B are test documents that are invalid. The Parser should signal some
error for each one of the test documents listed in Appendix B.
[/P0]
[/CHAP]
[CHAP]
[CHAPNO]5.2
[/CHAPNO]
[CHAPNM]Test Control
[/CHAPNM]
[P0]A list of all validation suite test documents is provided in Appendices
A and B. These test documents completely specify all input files necessary
to test that the EDS SGML Parser can correctly parse documents
that conform to the SGML standard and detect those documents that do not.
[/P0]
[/CHAP]
[CHAP]
[CHAPNO]5.3
[/CHAPNO]
[CHAPNM]Test Procedures
[/CHAPNM]
[P0]To run the Unit Test Plan, the symbol SGML must be correctly set up to
invoke execution of the program SGML.EXE. Assuming that the symbol is
correctly set, then the tester must execute
the following commands to parse a test document manually:
[/P0]
[LIST]
[I]$SET DEFAULT (directory containing the validation suite files)
[/I]
[I]$SGML (validation suite file name)
[/I]
[I]
[/I]
[/LIST]
[P0]To execute the entire validation suite using the command procedure
PARVAL.COM, the tester should execute the following commands:
```

```
[/PO]
[LIST]
[1]$SET DEFAULT    (directory containing the validation suite files)
[/1]
[1]$@PARVAL
[/1]
[1]$DIFF PARVAL.TST PARVAL.SAV
[/1]
[1]
[/1]
[/LIST]
[REPORT]
[REPBODY
    FILE='g.lis'
    FMT='TEXT']
[/REPBODY]
[/REPORT]
[REPORT]
[REPBODY
    FILE='i.lis'
    FMT='TEXT']
[/REPBODY]
[/REPORT]
[/CHAP]
[/SECT]
[/BODY]
[REAR]
[/REAR]
[/ICAMUTP]
```

APPENDIX C

EDS Unit Test Plan Source Document

The following pages are computer listings of an EDS source unit test plan manual.

Page 1 RANXER EDE PARER UTY PARER UTY 29 12-DEC-1987 10.22 - Page 001

P110 [REMOVED] EDE.PARSER.UT7;PARSER.UT7:25 12-DEC-1987 14:22 — Page 80.

File [REMOVED] EDS PARSED UTY PARSED UTY 25 12-DEC-1987 14:22 — Page 803

```

</item>
<body>
<sectionSECTION 14 Section>
<sectionSECTIONAL/Section>
<chap>
<chapno1 1/Chapter>
<chapno Purpose/Chapter>

<p>This Unit Test Plan (UTP) establishes the methodology and procedures
used to adequately test the capabilities of the computer program identified
as the BGC Parser. The BGC Parser is one configuration item of the
L2331 (L2331) ACIS (EDS) (p2)
</p>

<chap>
<chapno1.2/Chapter>
<chapno Project References/Chapter>

<list>
<listitem 1> Syntex (ul)ICAP Documentation Standards (ul)
EDS:30120000 5 September 1983 (ul)

<listitem 2> International Organization for Standardization
(ul)Information Processing - Text and Office Systems -
Standard Generalized Markup Language (SGML) (ul) ISO 8879
15 October 1986 (ul)

<listitem 3> International Organization for Standardization (ul)Office
Document Architecture/Office Document Interchange
Format (ul) ISO/DI 8632-1-4 October 1985 (Draft) (ul)

<listitem 4> American National Standards Institute (ul)American
National Standard for Information Systems - Computer
Graphics - Metafile for the Storage and Transfer of
Picture Description Information (ul) ANSI X3.127-1986
August 27, 1986 (ul)

<listitem 5> Structural Dynamics Research Corporation (ul)Form
Processor User's Manual (ul) UM 620244200A, 16 February 1987 (ul)

<listitem 6> Structural Dynamics Research Corporation (ul)Virtua
Terminal Operator Guide (ul) UM 620244000A, 16 February 1987 (ul)

<listitem 7> W. E. Lisk (ul)LEX - Lexical Analyzer Generator, IS
Workbench for VAX/VMS Programmers Guide (ul) (ul)

<listitem 8> Structural Dynamics Research Corporation (ul)Form
Processor Development Specifications (ul) DS 620244700A, 16 February 1987 (ul)
</list>

</chap>
<chap>
<chapno1 3/Chapter>
<chapno Terms and Abbreviations/Chapter>

<list>
<listitem 1> American Standard Code for Information Interchange (ASCII) (ul)
The character set defined by ANSI X3.4 and used by most computer
vendors (ul)

<listitem 2> Attribute (ul) A characteristic used to qualify an element
within a document (ul)

```

File [REMOVED] EDS PARSED UTY PARSED UTY 25 12-DEC-1987 14:22 — Page 803

File [REMOVED] EDS PARSED UTY PARSED UTY 25 12-DEC-1987 14:22 — Page 804

```

<listitem 3> Character Set (ul) A mapping of a character repertoire onto a
code set such that each character is associated with its coded
representation (ul)

<listitem 4> Compound Document (ul) A document which may contain mixed
content (text, graphics, etc.) (ul)

<listitem 5> Computer Graphics Metafile (CGM) (ul) A standard file format
for the storage and retrieval of picture description
information (ul)

<listitem 6> Computer Program Configuration Item (CPCI) (ul) An aggregation
of computer programs or any of their discrete portions, which
satisfies an end-use function (ul)

<listitem 7> Conforming BGC Application (ul) An BGC application that
requires documents to be conforming BGC documents, and whose
documentation meets the requirements of this International
Standard (ul)

<listitem 8> Context-Directed Editor (ul) An EDS application which guides
the user through the process of document creation and revision
by using the document type definition as a model for which
logical elements may be included in the document (ul)

<listitem 9> Descriptive Markup (ul) Information added to a document that
enables an application program to process the document (ul)

<listitem 10> Document Type Definition (DTD) (ul) Rules determined by an
application that apply BGC to the markup of documents of a
particular type. A document type definition includes a formal
specification, expressed in a document type declaration, of the
element types, element relationships and attributes, and
references that can be represented by markup. It therefore
defines the vocabulary of the markup for which BGC defines the
syntax. A DTD can also include comments that describe the
semantics of elements and attributes, and any application
conventions (ul)

<listitem 11> Electronic Documentation System (EDS) (ul) An integrated set
of software tools and application programs which operate upon a
document through various stages of a document life cycle
consisting of editing, proofreading, formatting, saving,
storage, and transferring (ul)

<listitem 12> Element (ul) A component of the hierarchical structure defined
by a document type definition. It is identified in a document
instance by descriptive markup, usually a start-tag and end-tag (ul)

<listitem 13> Element Declaration (ul) A markup declaration that contains
the formal specification of the part of an element type
definition that deals with the content and markup manipulation (ul)

<listitem 14> Entity (ul) A collection of characters that can be referenced
as a unit (ul)

<listitem 15> Entity Declaration (ul) A markup declaration that assigns an BGC name
to an entity so that it can be referenced (ul)

<listitem 16> Entity Reference (ul) A reference that is replaced by an entity (ul)

<listitem 17> Field (ul) Two-dimensional space on a terminal screen (ul)

<listitem 18> Form (ul) A structured view which may be imposed on windows or
other forms. A form is composed of fields. These fields may be

```

File [REMOVED] EDS PARSED UTY PARSED UTY 25 12-DEC-1987 14:22 — Page 804

File: [REDACTED] EDS PARSED UTY [PARSED UTY.25] 12-DEC-1987 14:22 -- Page: 605

(12)(ul)Form Definition (FD) (A1): Form definition language after compilation. It is read at run-time by the Form Processor. (A12)

(12)(ul)Form Definition Language (FDL) (A1): The language in which electronic forms are defined. (A12)

(12)(ul)Form Editor (FE) (A1): A subset of the IIS User Interface UI is used to create definitions of forms. The FE consists of the Form Driven Form Editor and the Form Language Compiler. (A12)

(12)(ul)Form Hierarchy (A1): A graphic representation of the way in which form items and windows are related to their parent form. (A12)

(12)(ul)Form Language Compiler (FLC) (A1): A subset of the FE that consists of a batch process that accepts a series of form definition language statements and produces form definition files as output. (A12)

(12)(ul)Form Processor (FP) (A1): A subset of the IIS User Interface that consists of a set of callable execution-time routines available to an application program for form processing. (A12)

(12)(ul)Form Driven Form Editor (FDFFE) (A1): A subset of the FE which consists of a form-driven application used to create Form Definition Files interactively. (A12)

(12)(ul)Generic Identifier (A1): A name that identifies the element type of an element. (A12)

(12)(ul)GI (A1): Generic Identifier. (A12)

(12)(ul)IIS Function Screen (A1): The first screen that is displayed after login. It allows the user to specify the function to access and the device type and device name on which to work. (A12)

(12)(ul)Integrated Information Support System (IIS) (A1): A test computing environment used to investigate, demonstrate, and test the concepts of information management and information integration in the context of Aerospace Manufacturing. The IIS addresses the problems of integration of data resident on heterogeneous data bases supported by heterogeneous computers interconnected via a Local Area Network. (A12)

(12)(ul)Item (A1): A non-decomposable area of a form in which hard-coded descriptive text may be placed and the only defined areas where user data may be input/output. (A12)

(12)(ul)Layout Style (A1): The specification of format and presentation for logical elements. (A12)

(12)(ul)Layout Structure (A1): The hierarchy of all layout elements (pages, frames, blocks, etc.) for a document. (A12)

(12)(ul)Logical Structure (A1): The hierarchy of all logical elements (paragraphs, sections, etc.) within a document. (A12)

(12)(ul)Markup (A1): Text that is added to the data of a document in order to convey information about it. (A12)

(12)(ul)Markup Manipulation (A1): A feature of SOPE that allows markup to be manipulated by aborting or omitting tags, or shortening entity references. (A12)

File: [REDACTED] EDS PARSED UTY [PARSED UTY.25] 12-DEC-1987 14:22 -- Page: 605

File: [REDACTED] EDS PARSED UTY [PARSED UTY.25] 12-DEC-1987 14:22 -- Page: 606

standard message line on the terminal screen. Messages are used to warn of errors or provide other user information. (A12)

(12)(ul)Message Line (A1): A line on the terminal screen that is used to display messages. (A12)

(12)(ul)Operating System (OS) (A1): Software supplied with a computer which allows it to supervise its own operations and manage access to hardware facilities such as memory and peripherals. (A12)

(12)(ul)Page (A1): Instance of form in window that are created whenever a form is added to a window. (A12)

(12)(ul)Paging and Scrolling (A1): A method which allows a form to contain more data than can be displayed at one time with provisions for viewing any portion of the data buffer. (A12)

(12)(ul)Parser (A1): An application program that determines how closely a document conforms to a document type definition which defines a specific documentation standard. (A12)

(12)(ul)Physical Device (A1): A hardware terminal. (A12)

(12)(ul)Previous Cursor Position (A1): The position of the cursor when the previous edit command was issued. (A12)

(12)(ul)Qualified Name (A1): The name of a form, item, or window preceded by the hierarchy path so that it is uniquely identified. (A12)

(12)(ul)Standard Generalized Markup Language (SGML) (A1): A language for describing document structures, consisting of descriptive markup which is added to a document to indicate where logical elements such as sections and paragraphs begin and end. (A12)

(12)(ul)Subform (A1): A form that is used within another form. (A12)

(12)(ul)Tag (A1): Descriptive markup indicating the start or end of a logical element. (A12)

(12)(ul)Tagger (A1): An application program which provides a mechanism for automatically tagging existing documents which have been created by word processing systems. (A12)

(12)(ul)User Interface (UI) (A1): IIS subsystem that controls the user's terminal and interfaces with the rest of the system. The UI consists of two major subsystems: The User Interface Development System (UIDS) and the User Interface Management System (UIMS). (A12)

(12)(ul)User Interface Management System (UIMS) (A1): The run-time UI. It consists of the Form Processor, Virtual Terminal, Application Interface, the User Interface Services, and the Text Editor. (A12)

(12)(ul)User Interface Services (UIS) (A1): A subset of the IIS User Interface that consists of a package of routines that aid users in controlling their environment. It includes message management, change password, and application definition services. (A12)

(12)(ul)User Interface/Virtual Terminal Interface (UI/VTI) (A1): Another name for the User Interface. (A12)

(12)(ul)Virtual Terminal (VT) (A1): A subset of the IIS User Interface that performs the interfacing between different terminals and

File: [REDACTED] EDS PARSED UTY [PARSED UTY.25] 12-DEC-1987 14:22 -- Page: 606

File: [REDACTED] EDS_PARSER.UTP/PARSER.UTP.25 12-DEC-1987 14:22 -- Page: 007

features and protocols which must be supported by the UT software which constitutes the virtual terminal definition. Specific terminals are then mapped against the virtual terminal software by specific software modules written for each type of real terminal supported. (/12)

(12)Virtual Terminal Interface (VTI) (/12) The callable interface to the VTI. (/12)

(12)Window (/12) Dynamic area of a terminal screen on which predefined forms may be placed at run-time. (/12)

(12)Window Manager (/12) A facility which allows the following to be manipulated: size and location of windows, the device on which an application is running, the position of a form within a window. It is part of the Form Processor. (/12)

(/chap)

(/sect)

(sect)

(sect)SECTION 2 (/sect)

(sect)DEVELOPMENT ACTIVITY (/sect)

(chap)

(chap)1 (/chap)

(chap)Statement of Project Activities (/chap)

(p)During system development, the computer programs were tested progressively. Functionality was incrementally tested, and as bugs were discovered by this testing, the software was corrected. (/p)

(p)The starting point for the development of the EDS SGC Parser was a Parser Implementation for the M-DCS operating system developed by the National Bureau of Standards (NBS) as part of the CACS initiative. For EDS, the Parser was ported from M-DCS to the operating systems supported by IIS. As the code was ported, changes were made to eliminate the use of temporary files, fix minor bugs, and add additional functionality needed to support EDS. (/p)

(p)The NBS also developed an SGC Parser validation suite consisting of a number of files that test specific functionality outlined in the SGC standard (SC 8879). The validation suite will serve as the basis for the EDS SGC Parser Unit Test Plan. (/p)

(/chap)

(chap)

(chap)2 (/chap)

(chap)Project Activity Results (/chap)

(p)The EDS SGC Parser successfully passed the NBS validation suite. A few minor errors were found in the validation suite files themselves and these were corrected. (/p)

(/chap)

(/sect)

(sect)

(sect)SECTION 3 (/sect)

(sect)SYSTEM DESCRIPTION (/sect)

File: [REDACTED] EDS_PARSER.UTP/PARSER.UTP.25 12-DEC-1987 14:22 -- Page: 007

File: [REDACTED] EDS_PARSER.UTP/PARSER.UTP.25 12-DEC-1987 14:22 -- Page: 008

(chap)1 (/chap)

(chap)System Description (/chap)

(p)The SGC Parser is the component of the Electronic Documentation System that validates the logical structure of a document against a previously built SGC Document Type Definition. It insures that the descriptive markup in the document conforms to the set of rules defined by the DTD. The DTD data defines what generic identifiers are valid at any point in the document and how many times they may occur. (/p)

(p)In addition to validating the markup, the Parser also converts the document to its fully marked up state by processing all Entity References, Attributes, and expands all unescaped generic identifiers found within the document. (/p)

(/chap)

(chap)

(chap)3 (/chap)

(chap)Testing Schedule (/chap)

(p)Since the NBS validation suite is the basis for the Parser test plan, execution of this unit test plan is not dependent upon any other components of either EDS or IIS. (/p)

(/chap)

(chap)

(chap)3 (/chap)

(chap)First Location Testing (/chap)

(p)These tests of the SGC Parser require the following: (/p)

(list)

(1)Equipment: All Force VAX terminals supported by the Virtual Terminal (/1)

(2) (/1)

(1)Support Software: C run-time libraries (/1)

(2) (/1)

(1)Personnel: one integrator familiar with EDS (/1)

(2) (/1)

(1)Training: the EDS User Manual has been previously delivered (/1)

(2) (/1)

(1)Deliverables: the SGC Parser CPC (/1)

(2) (/1)

(1)Test Materials: All tests are run using the NBS SGC validation suite files (/1)

(2) (/1)

(1)Security Considerations: None (/1)

(2) (/1)

(/list)

(/chap)

(chap)

(chap)3 (/chap)

(chap)Subsequent Location Testing (/chap)

(p)The requirements listed above must be met. The command procedure

File: [REDACTED] EDS_PARSER.UTP/PARSER.UTP.25 12-DEC-1987 14:22 -- Page: 008

File: [REDACTED] EDS_PARSER_UTP\PARSER_UTP.25 12-DEC-1987 14:22 - Page: 009

```

The output file PARVAL.TXT can then be compared using the VAX DIFF
command to the file PARVAL.SAV under IIS5 ON to insure that the unit
test ran correctly. (/p0)

</chap>

</sect>

</sect>

<sect>
<sect>SECTION 4</sect>
<sect>SPECIFICATIONS AND EVALUATIONS</sect>
</sect>

<chap>
<chap>1</chap>
<chap>Test Specification</chap>

<p0>The Unit Test Plan is based on covering specific functionality of the
SGPL Parser as outlined in the EDS Development Specification (ES) and the
SGPL IIS 87's reference manual. (/p0)

<p0>The Parser is tested by using the input files of the EDS validation
suite. These files were developed to test the level of conformance
and the correctness of Parser implementations to the SGPL standard.
Each validation suite test file tests one specific feature of the standard.
The first set of files, listed in Appendix A and starting with the letter
G, tests sequences of SGPL language statements that are valid. No errors
should be detected by the Parser when parsing any of the valid test cases.
The second set of files, listed in Appendix B and starting with the letter
I, are test sequences of SGPL language statements that are invalid
and should produce an error when they are parsed. To execute the validation
suite, each file can be parsed individually by the tester, or a command
procedure PARVAL.COM can be executed to parse all files at once. (/p0)

<p0>The validation suite is the Unit Test for the EDS SGPL Parser. The
objective of the test is to insure that the SGPL Parser parses all
validation suite files in the correct manner. (/p0)

</chap>

<chap>
<chap>2</chap>
<chap>Testing Methods and Constraints</chap>

<p0>The tests outlined in Section 5 can be executed in any order. The required
input is given for each test in the form of a validation suite test file name.
A list of the test files are given
in Appendices A & B. Appendix A contains those tests that should run
without error while Appendix B contains those tests that should produce
an error. (/p0)

<p0>The tester can either manually parse one file at a time, or run a command
file to execute the entire validation suite. The name of this
command file is PARVAL.COM and is under IIS5 Configuration Management. (/p0)

<p0>No additional constraints are placed on this unit test
besides those listed in Sections 5.2 and 5.3 of this document. (/p0)

</chap>

<chap>
<chap>3</chap>
<chap>Test Progression</chap>

<p0>The validation suite test files may be executed in any order. (/p0)

```

File: [REDACTED] EDS_PARSER_UTP\PARSER_UTP.25 12-DEC-1987 14:22 - Page: 009

File: [REDACTED] EDS_PARSER_UTP\PARSER_UTP.25 12-DEC-1987 14:22 - Page: 010

```

</chap>

<chap>
<chap>4</chap>
<chap>Test Evaluation</chap>

<p0>The EDS validation suite test files are divided into two categories -
test documents that should parse without error, and test documents that
are invalid and should produce an error. The files that begin with the letter
G should NOT produce any errors. The files that begin with the letter I
SHOULD produce an error message. (/p0)

<p0>The test results are evaluated by checking the Parser output to make sure
that no error messages have been generated for G files and that error
messages are generated for all I files. (/p0)

<p0>The SGPL Parser will stop parsing the test document and generate an error
message if the term ERROR is found in the document. The
accuracy of this Unit Test Plan is dependent upon the observation by the tester
of the output of the Parser. (/p0)

<p0>If the tester runs PARVAL.COM then the output file PARVAL.TXT is
produced. This file may be compared to the file PARVAL.SAV under IIS5
Configuration Management using the VAX DIFF command. For the test
to be successful, the files should match exactly. (/p0)

</chap>

</sect>

</sect>

<sect>
<sect>SECTION 5</sect>
<sect>TEST PROCEDURES</sect>
</sect>

<chap>
<chap>1</chap>
<chap>Test Description</chap>

<p0>The Unit Test plan is executed by parsing all test documents listed in
Appendices A and B and observing the output of the Parser for each test.
As mentioned above, the test documents listed in Appendix A are valid
SGPL documents and should parse without error. Those listed in Appendix
B are test documents that are invalid. The Parser should signal some
error for each one of the test documents listed in Appendix B. (/p0)

</chap>

<chap>
<chap>2</chap>
<chap>Test Control</chap>

<p0>A list of all validation suite test documents is provided in Appendices
A and B. These test documents completely specify all input files necessary
to test that the EDS SGPL Parser can correctly parse documents
that conform to the SGPL standard and detect those documents that do not. (/p0)

</chap>

<chap>
<chap>3</chap>
<chap>Test Procedures</chap>

<p0>To run the Unit Test Plan, the symbol SGPL must be correctly set up to

```

File: [REDACTED] EDS_PARSER_UTP\PARSER_UTP.25 12-DEC-1987 14:22 - Page: 010

File: [REDACTED] EDE.PARSER.UTP\PARSER.UTP.25 22-DEC-1987 14:22 -- Page: 011

correctly set, then the tester must execute the following commands to parse a test document manually: (/pc)

```
(list)
(1)SET DEFAULT (directory containing the validation suite files)(/1)
(1)SOPC (validation suite file name)(/1)
(1) (/1)
(/list)
```

(pc)To execute the entire validation suite using the command procedure PARVAL.CPP, the tester should execute the following commands (/T):

```
(list)
(1)SET DEFAULT (directory containing the validation suite files)(/1)
(1)SPARVAL(/1)
(1)SETTY PARVAL.TTY PARVAL.SAV(/1)
(1) (/1)
(/list)
```

```
(report)
(1)report, file="g.lis"
(/report)
(/report)
```

```
(report)
(1)report, file="i.lis"
(/report)
(/report)
```

```
(/chap)
(/sect)
```

```
(/end)
```

```
(/end)
(/foot)
(/icamitp)
```

File: [REDACTED] EDE.PARSER.UTP\PARSER.UTP.25 22-DEC-1987 14:22 -- Page: 011

APPENDIX D

DECDX

The following pages are computer listings of EDS Tagger information. The first listing, DECDX.L, is that of the template for all DECDX documents to be tagged using the Tagger facilities. The second listing is a listing that was generated using the DECDX.L as a template and running the Autotag program.

File DECDX.L

This file is the template for all DECDX documents to be tagged.

```
%p 3000
%|
/* DECDX - DEC DX document template
 *
 * Description
 * This lex source file is used as a template for DEC DX
 documents. It
 * would be loaded at the start of creating a tagger for a
 particular
 * document and saved under a different name relevant to the
 document.
 */
#include <stdlib.h>
#include <stdio.h>
#include <ctype.h>
#include <ctype.h>

#undef input
#undef unput
#undef YYLMAX
#undef ECHO

#define YYLMAX 10000
#define ECHO echo(yytext)
#define EMIT(x) fprintf(outfp, "%s", x)
#define yywrap() 1

static yylook(), yyback(), yyinput(), yyoutput(), yyunput();
extern FILE *outfp;
%}
bold_on          "{s"
bold_off         "{\\"
und_on           "{s"
und_off          "{S"
sup_on           "{)"
sup_off          "{("
sub_on           "{+"
sub_off          "{-"
aux_on           "{."
aux_off          "{;"
dx_tilde         ">"
dx_l_brace       ">:"
dx_r_brace       "<:"
dx_bar           "<"
lin_mod          "|G"
com_on           "|H"
dx_tab           "|I"
dx_nl            "|J"
```

```

dx_ff          "L"
com_off        "M"
space          "-"
hyphen         "-"
vw_hyphen      {aux_on}{und_on}{dx_n1}
vw_eol         {aux_on}{dx_n1}{aux_off}
page_mark      {aux_on}{dx_ff}{aux_off}
breaking_hyphen {aux_on}{hyphen}{aux_off}
paragraph_mark {sup_on}{dx_n1}{sup_off}
center         {sub_on}{dx_n1}{sub_off}
ruler_modified {bold_on}{lin_mod}{bold_off}
tde_on         {com_on}{space}"5" {com_off}
tde_off        {com_on}{space}"55" {com_off}
tde_char       {com_on}{space}4. {com_off}
any_char
{[ ]|[ ]|[dx_tab]|[dx_bar]|[dx_l_brace]|[dx_r_brace]|[dx_tilde]}
white_space    ([space] [dx_tab])
}
{
}
/* anything else */
if (yytext[0] == '|' || yytext[0] == '(') yymore();
else ECHO;
}
}
static char buf[YYLMAX], *pbuf = buf;
int charyy = 0;
static input()
{
    char c;

    if (pbuf > buf) c = *pbuf--;
    else c = zzlex();
    return c;
}
static unput(c)
char c;
{
    *++pbuf = c;
}
static yyreject()
{
    extern int yylength;

    while (yylength > 0)
        *++pbuf = yytext[--yylength];
}

```

File DXUTP.L

This file is the result of running the AUTOTAG program, steps 2 through 32 of this unit test plan.

```
%p 3000
%{
/* DECDX - DEC DX document template
 *
 * Description
 *   This lex source file is used as a template for DEC DX
 documents. It
 *   would be loaded at the start of creating a tagger for a
 particular
 *   document and saved under a different name relavent to the
 document.
 */
#include <stdtyp.h>
#include <stdio.h>
#include <ctlchr.h>
#include <ctype.h>

#undef input
#undef unput
#undef YYLMAX
#undef ECHO

#define YYLMAX 10000
#define ECHO echo(yytext)
#define EMIT(x) fprintf(outfp, "%s", x)
#define yywrap() 1

static yylook(), yyback(), yyinput(), yyoutput(), yyunput();
extern FILE *outfp;
%}

bold_on      "{s"
bold_off     "{\\"
und_on       "{s"
und_off      "{S"
sup_on       "{)"
sup_off      "{("
sub_on       "{+"
sub_off      "{-"
aux_on       "{'"
aux_off      "{t"
dx_tilde     ">"
dx_l_brace   "|:"
dx_r_brace   "|-"
dx_bar       "<"
lin_mod      "|C"
com_on       "|H"
dx_tab       "|I"
```

```

dx_nl      "|J"
dx_ff      "|L"
com_off    "|M"
space      "-"
hyphen     "-_"
ww_hyphen  {aux_on}{und_on}{dx_nl}
ww_eol     {aux_on}{dx_nl}{aux_off}
page_mark  {aux_on}{dx_ff}{aux_off}
breaking_hyphen {aux_on}{hyphen}{aux_off}
paragraph_mark {sup_on}{dx_nl}{sup_off}
center     {sub_on}{dx_nl}{sub_off}
ruler_modified {bold_on}{lin_mod}{bold_off}
tde_on     {com_on}{space}"#5"{com_off}
tde_off    {com_on}{space}"#55"{com_off}
tde_char   {com_on}{space}4.{com_off}
any_char   " "
{[|!|]{dx_tab}{dx_bar}{dx_l_brace}{dx_r_brace}{dx_tilde}}
white_space {space}{dx_tab}
snum        {[0-9]+("."[0-9]+)+}
dnur        {[0-9]+("-"[0-9]+)+}
number      {[0-9]+}
format      {([und_on]{und_off}){bold_on}{bold_off}{com_on}{com_off}}
%start BODY
%start CHAPNUM
%start READNL
%start FIGNUM
%start FIGTITLE
%start FIGEND
%start SECTITLE
%%
%{
BEGIN BODY;
%}
<BODY>{und_on}?SECTION{space}*{number}{und_off}?/{center} {
/* a section header */
EMIT("<sectnum>");
ECHO;
EMIT("</sectnum>");
BEGIN SECTITLE;
}
<SECTITLE>{any_char}{center}+/{dx_nl}{dx_nl} {
/* section name */
EMIT("<secttitle>");
ECHO;
EMIT("</secttitle>");
BEGIN READNL;
}
<BODY>{snum}{space}*/{und_on}?{any_char}{und_off}?{dx_nl}{dx_nl}
} {
/* a numbered paragraph */
EMIT("<chapnum>");
ECHO;
EMIT("</chapnum>");

```

```

BEGIN CHAPNUM;
}
<CHAPNUM>{und_on}?{any_char}+{und_off}?/{dx_nl}{dx_nl} {
/* paragraph name */
EMIT("<chptitle>");
ECHO;
EMIT("</chptitle>");
BEGIN READNL;
}
<BODY>({any_char}|{vw_eol}|{format})+/{dx_nl}{white_space}*{dx_n
l} {
/* a paragraph */
EMIT("<para0>");
ECHO;
EMIT("</para0>");
BEGIN READNL;
}
<READNL>{dx_nl}{white_space}*{dx_nl} {
/* eat new lines after paragraphs and headers */
EMIT("\n");
BEGIN BODY;
}
<BODY>(({any_char}|{format})+{dx_nl})+{dx_nl} {
/* a figure, maybe */
EMIT("<figbody>");
ECHO;
EMIT("</figbody>");
}
<BODY>"Figure"{white_space}+/{dnum}{white_space}+{any_char}+{cen
ter} {
/* figure title */
EMIT("<figref>");
ECHO;
EMIT("</figref>");
BEGIN FIGNUM;
}
<FIGNUM>{dnum}{white_space}+ {
/* figure number */
EMIT("<fignum>");
ECHO;
EMIT("</fignum>");
BEGIN FIGTITLE;
}
<FIGTITLE>{any_char}+ {
/* figure name */
EMIT("<figtitle>");
ECHO;
EMIT("</figtitle>");
BEGIN FIGEND;
}
<FIGEND>{center}({white_space}*{dx_nl})* {
/* figure cleanup */
EMIT("\n");
BEGIN BODY;

```