NOT
MEASUREMENT
SENSITIVE

MIL-PRF-28001C
2 MAY 1997

SUPERSEDING
MIL-PRF-28001B
AMENDMENT 1
20 APRIL 1995

PERFORMANCE SPECIFICATION

MARKUP REQUIREMENTS AND GENERIC STYLE SPECIFICATION FOR EXCHANGE OF TEXT AND ITS PRESENTATION

This specification is approved for use by all Departments and Agencies of the Department of Defense.

Beneficial comments (recommendations, additions, deletions) and any pertinent data which may be of use in improving this document should be addressed to: ATTN: CALS Digital Standards Office, DISA Center for Standards (CFS), Code JIEO/JEBEB, 10701 Parkridge Blvd, Reston, VA, 20191-4357 by using the Standardization Document Improvement Proposal (DD Form 1426) appearing at the end of this document or by letter.

AMSC N/A AREA IPSC

1 SCOPE

1.1 SCOPE. To set forth the overall requirements for the creation and management of Standard Generalized Markup Language (SGML) applications. This specification defines performance characteristics for the following SGML applications:

a. Document type declaration.
b. Document Type Definition (DTD).
c. Document instance.
d. Formatting Output Specification Instance (FOSI).

1.2 APPLICABILITY. SGML shall be used:

a. To describe the logical structure of textual information in an unambiguous grammar.
b. To describe the presentation of textual information in an unambiguous grammar.
c. To assure automated quality control over adherence to that structure.
d. To deliver and store textual information in an easily maintained and updated form.

2 APPLICABLE DOCUMENTS

2.1 GENERAL. The documents listed in this section are specified in sections 3 and 4 of this specification. This section does not include documents cited in other sections of this specification or recommended for additional information or as examples. While every effort has been made to ensure the completeness of this list, document users are cautioned that they must meet all specified requirements documents cited in sections 3 and 4 of this specification, whether or not they are listed.

2.2 GOVERNMENT DOCUMENTS.

2.2.1 Specifications, standards, and handbooks. The following specifications, standards, and handbooks form a part of this document to the extent specified herein. Unless otherwise specified, the issues of these documents are those listed in the issue of the Department of Defense Index of Specifications and Standards (DoDISS) and supplement thereto, cited in the solicitation (see 6.2).

STANDARDS

DEPARTMENT OF DEFENSE

MIL-STD-1840 - Automated Interchange of Technical Information

(Unless otherwise indicated, copies of the above specifications, standards, and handbooks are available from Standardization Document Order Desk, 700 Robbins Avenue, Building 4D, Philadelphia, PA 19111-5094.)

2.2.2 Other Government documents, drawings, and publications. The following other Government documents, drawings, and publications form a part of this document to the extent specified herein. Unless otherwise specified, the issues are those cited in the solicitation.

CSL Procedures - CSL Construct and Object Submission Procedures

(Copies of the above document may be obtained from the CALS Digital Standards Office, DISA, Center for Standards, Code JIEO/JEBEB, 10701 Parkridge Blvd, Reston, VA 20191–4357.)

2.3 NON-GOVERNMENT PUBLICATIONS. The following document(s) form a part of this document to the extent specified herein. Unless otherwise specified, the issues of the documents which are Department of Defense (DoD) adopted are those listed in the issue of the DoDISS cited in the solicitation. Unless otherwise specified, the issues of documents not listed in the DoDISS are the issues of the documents cited in the solicitation (see 6.2).

INTERNATIONAL ORGANIZATION FOR STANDARDIZATION (ISO)

ISO 8879 - Information processing — Text and office systems — Standard Generalized Markup Language (SGML) (DoD adopted)

(Application for copies should be addressed to the Standardization Document Order Desk, 700 Robbins Ave., Building 4D, Philadelphia, PA 19111-5094, for issue to DoD

activities only. All other requestors must obtain documents from the American National Standards Institute, 11 West 42nd Street, 13th Floor, New York, NY 10036.)

2.4 ORDER OF PRECEDENCE. In the event of a conflict between the text of this document and the references cited herein, the text of this document takes precedence. Nothing in this document, however, supersedes applicable laws and regulations unless a specific exemption has been obtained.

3 REQUIREMENTS

3.1 GENERAL. This specification defines the requirements for the preparation of items a through f. The contract or other agreement specifies which of the item(s) shall be prepared (see 6.2).

a. SGML Declaration (see 3.9).
b. Document type declaration (see 3.2).
c. Document type definition (see 3.3).
d. Document instance (see 3.4).
e. FOSI (see 3.5).
f. Other mechanism(s) for preparing interchangeable style and format information shall be specified in accordance with the contract or other form of agreement.

3.2 DOCUMENT TYPE DECLARATION. The document type declaration shall conform to ISO 8879, this specification, and reference a contractually specified DTD with a Formal Public Identifier (FPI) (see 3.3.3).

3.3 DOCUMENT TYPE DEFINITION (DTD). A DTD shall conform to ISO 8879, the SGML Declaration in 3.9 (or an acceptable modification thereof), and the detail requirements specified in the contract or other form of agreement (see 6.2).

3.3.1 Preparation for reuse. When specified in the contract (see 6.2), the DTD shall contain DoD standard SGML objects (elements, attributes, and entities) as defined in the CALS SGML Library (CSL).

3.3.2 SGML object registration. When the definition of the document's structure/content requires SGML objects not in the CSL, these objects shall be submitted to the CSL for approval using the CSL Construct and Object Submission Procedures as specified by contract or other form of agreement (see 6.2).

3.3.3 Formal Public Identifier (FPI). A completed DTD shall have an FPI conforming to ISO 8879 and this specification. The FPI specifically defines the version of a completed DTD. An FPI shall not identify more than one DTD or more than one version of a DTD. FPIs such as " `-//USA-DOD//DTD EXAMPLE MIL-HDBK-28001//EN`" shall have the following two parts, separated by a double slash ("`//`"):

a. An owner identifier. For all DoD components, this shall be " `-//USA-DOD`" entered as shown without the quotation marks.
b. A minimal description (called the "text identifier" in ISO 8879). This description is divided into three fields, the first and second separated by a single space character and the second and third separated by a double slash ("`//`"):
    1. Public text class – This is the name of an SGML construct listed in ISO 8879, in the example, this is "`DTD`."
    2. Public text description – A short description of the construct being identified. This description should include the revision status (for example, "REV B"),

if any, and the date of release to ensure complete identification of the construct. The date shall be provided in the format CCYYMMDD. An example of such a public text description for a hypothetical DTD would be: `MIL-PRF-1234 REV B 19960123`

3. Public text language — A two character language code, in the example, this is "`EN`".

Note that, in public identifiers, letter case and the presence of spaces and record ends are significant. More precisely, a record end is treated as a space, and then a run of consecutive spaces is treated as a single space; therefore, while a record end and extra spaces can be inserted wherever a single space is shown, neither a record end nor a space may be inserted where no space is shown, including immediately before or after the double slashes.

3.4 DOCUMENT INSTANCE. The document instance shall conform to ISO 8879, the SGML declaration in 3.9 (or an acceptable modification thereof), and to the contractually specified DTD.

3.5 FORMATTING OUTPUT SPECIFICATION INSTANCE (FOSI). A FOSI shall conform to the Output Specification (OS) in appendix B and to ISO 8879. A FOSI shall specify a layout object for each unique page, frame, or screen. Layout object characteristics shall be as specified in the contract or other form of agreement. Presentation characteristics of every displayable element declared in the source DTD shall be specified in the FOSI. Values for presentation characteristics shall be as specified in the contract (see 6.2). The inheritance of presentation characteristics for elements within a content model of the source DTD shall be as specified in the OS.

3.6 NEW DTDS OR FOSIS. Unless otherwise specified in the contract or other form of agreement (see 6.2), the development of program-specific DTDs or FOSIs is discouraged.

3.7 NOTATION DECLARATIONS. Unless otherwise specified, notation declarations used in DTDs or FOSIs developed in accordance with this specification shall be those contained in the appropriate detail specification (see 6.2).

3.8 CONFORMANCE. When required by the contract or other form of agreement (see 6.2), each SGML document shall be subjected to conformance inspection in accordance with 4.1.

3.9 SGML DECLARATION. The following SGML Declaration declares the character set, syntax, quantities, capacities, scope, and features of SGML. Unless otherwise specified, this declaration shall be used when interchanging SGML documents under this specification. The quantities and capacities have been increased from the reference quantity and capacity sets in ISO 8879 to enable most DoD DTDs and tagged instances to parse without errors or warnings. If still larger quantities or capacities are required, the declaration quantities or capacities may be increased (see 6.2). The features in this declaration shall not be changed. If the declaration is modified, the modified declaration shall be included as part of the MIL-STD-1840 SGML Transfer Unit.

The SUBDOC feature has been set to “YES” in this declaration. SUBDOC is an optional feature within SGML and not all systems support SUBDOC. If SUBDOC is used, the declaration will not allow more than two subdocuments to be opened at the same time.

3.9.1 The SGML Declaration.

```
<!SGML "ISO 8879:1986"
CHARSET
BASESET  "ISO 646:1983//CHARSET International Reference Version (IRV)//ESC 2/5 4/0"

DESCSET                0       9      UNUSED
                       9       2        9
                      11       2      UNUSED
                      13       1       13
                      14      18      UNUSED
                      32      95       32
                     127       1      UNUSED

BASESET  "ISO Registration Number 100//CHARSET
                   ECMA-94 Right Part of Latin Alphabet Nr. 1//ESC 2/9 4/1"

DESCSET              128      32      UNUSED
                     160       5        32
                     165       1      UNUSED
                     166      88        38
                     254       1       127
                     255       1      UNUSED

CAPACITY SGMLREF
         TOTALCAP              1000000
         ENTCAP                 300000
         ELEMCAP                300000
         GRPCAP                 300000
         EXGRPCAP               300000
         EXNMCAP                300000
         ATTCAP                 300000
         AVGRPCAP               300000
         IDCAP                  300000
         IDREFCAP               300000
SCOPE                 DOCUMENT
SYNTAX

         SHUNCHAR CONTROLS 0 1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17
                     18 19 20 21 22 23 24 25 26 27 28 29 30 31 127 255
BASESET "ISO 646:1983//CHARSET
                      International Reference Version (IRV)//ESC 2/5 4/0"

DESCSET        0              128                    0
```

```
FUNCTION        RE                                  13
                RS                                  10
                SPACE                               32
                TAB                 SEPCHAR          9
NAMING          LCNMSTRT            ""
                UCNMSTRT            ""
                LCNMCHAR            "-."
                UCNMCHAR            "-."
                NAMECASE            GENERAL              YES
                                    ENTITY               NO
DELIM           GENERAL             SGMLREF
                SHORTREF            NONE
                NAMES               SGMLREF

QUANTITY    SGMLREF             ATTCNT                   400
                                ATTSPLEN                960000
                                ENTLVL                    1600
                                GRPCNT                     320
                                GRPGTCNT                   960
                                GRPLVL                    1600
                                LITLEN                  240000
                                NAMELEN                     32
                                TAGLEN                  960000
                                TAGLVL                     240

FEATURES
MINIMIZE    DATATAG    NO    OMITTAG    YES    RANK  NO    SHORTTAG    NO
LINK        SIMPLE     NO    IMPLICIT    NO    EXPLICIT    NO
OTHER       CONCUR     NO    SUBDOC     YES 2  FORMAL      YES
APPINFO      NONE >
```

4 VERIFICATION

4.1 CONFORMANCE. When required by the contract or other form of agreement, each SGML document developed in accordance with this specification shall meet the requirements in section 3. Additionally, each SGML document shall be parsed using at least three ISO 8879 compliant validating parsers. Each parser shall use the character set, capacities, quantities, and syntax specified in the applicable SGML Declaration. The contract or other form of agreement may specify additional quality or conformance requirements (see 6.2).

5 PACKAGING

5.1 PACKAGING. For acquisition purposes, the packaging requirements shall be as specified in the contract or order (see 6.2). When actual packaging of material is to be performed by DoD personnel, these personnel need to contact the responsible packaging activity to ascertain requisite packaging requirements. Packaging requirements are maintained by the Inventory Control Point's packaging activity within the Military Department or Defense Agency, or within the Military Department's System Command. Packaging data retrieval is available from the managing Military Department's or Defense Agency's automated packaging files, CD-ROM products, or by contacting the responsible packaging activity.

6 NOTES

(This section contains information of a general or explanatory nature that may be helpful, but is not mandatory.)

6.1 INTENDED USE. The use of DTDs and FOSIs will allow preparation of documents in an automated support environment using any or all of the following processes:

a. Creating a document type declaration or DTD for publication control if one does not already exist.
b. Creating a FOSI, if one does not already exist, to specify the formatting to be applied to documents conforming with the document type declaration.
c. Authoring the publication and inserting SGML markup tags.
d. Verifying that the syntax is correct according to the rules of SGML.
e. Using a FOSI and a document type declaration to direct the composition of the document so that the produced (printed or displayed) copy corresponds to the proper format and style.
f. Optionally, reviewing the document electronically using SGML for the comments.
g. Optionally, generating a text presentation metafile in a page description language (PDL) to drive the display device, such as a printer or typesetter.

6.2 ACQUISITION REQUIREMENTS. Acquisition documents for SGML developed in accordance with this specification must specify the following:

a. Title, number, and date of the document.
b. Issue of the DoDISS to be cited in the solicitation, and if required, the specific issue of individual documents referenced (see 2.2.1, 2.2.2, 2.3).
c. Packaging requirements (see 5.1).
d. The item(s) to be prepared in accordance with this specification (see 3.1).
e. Whether or not to use standard SGML objects from the CSL (see 3.3.1).
f. Whether or not new SGML objects will be submitted to the CSL for registration and inclusion in the CSL (see 3.3.2).
g. Values of presentation characteristics (see 3.5).
h. Authorization to develop program-specific DTDs or FOSIs (see 3.6)
i. Use of notation declarations not in a detail specification (see 3.7).
j. Whether or not to require a conformance inspection or other qualification requirements (see 3.8 and 4.1).

Appendix A provides additional information to assist acquisition personnel in determining the requirements and options that may need to be placed in the contract or other form of agreement.

6.3 TAILORING AND IMPLEMENTATION GUIDANCE. DTD developers should exercise care in their use of tailoring and implementation guidance. It is in the interest of both

DoD and industry to agree on the most widely applicable set of conventions for the preparation and interchange of technical publications. Implementation of these conventions should be left to the implementation system.

6.4 APPLICATION OF NON-GOVERNMENT STANDARDS. Current national and international non-Government standards do not adequately address all seven steps of the publication preparation process (see 6.1). ISO 8879 addresses steps a and d. ISO 10180 supports step g. No approved national or international standards exist to support steps b and e (FOSI), however, ISO 10179 covers the Document Style Semantics and Specification Language (DSSSL) which does define composition and format.

As work matures in these areas, this specification may be extended to define their implementation and application within DoD. In the interim, appendix B of this specification support steps b and e of the publication preparation process listed in 6.1.

6.5 DEFINITIONS.

6.5.1 Acronyms. Acronyms used in this specification are defined as follows:

CALS	Continuous Acquisition and Life-Cycle Support
CSL	CALS SGML Library
DoD	Department of Defense
DoDISS	Department of Defense Index of Specifications and Standards
DSSSL	Document Style Semantics and Specification Language
DTD	Document Type Definition
FOSI	Formatting Output Specification Instance
FPI	Formal Public Identifier
ISO	International Organization for Standardization
OS	Output Specification
PDL	Page Description Language
SGML	Standard Generalized Markup Language

6.5.2 Glossary. These definitions are for terms found in this specification and are consistent with those in ISO 8879. For a complete glossary of terms as applied to SGML use in DoD, see MIL-HDBK-28001. For the full set of formal SGML definitions, see ISO 8879.

6.5.2.1 Document type declaration. A markup declaration that contains the formal specifications of a document type definition, shown as:

```
<!DOCTYPE document_type_name optional_external_identifier [
 optional_document_type_declaration_subset ]>
```

6.5.2.2 Document Type Definition (DTD). An abstract collection of rules, determined by an application, that apply SGML to the markup of documents of a particular type.

(NOTE: “DTD” is occasionally—but not in compliance with ISO 8879 terminology—used as an abbreviation for “document type declaration”; it is also an SGML reserved word used in FPIs to indicate that the identified entity is a document type declaration set, and is often used to identify such a set.)

6.5.2.3 Formatting Output Specification Instance (FOSI). An instance of the OS that assigns values to the style characteristics for a particular document type declaration. The FOSI uses the syntax of an SGML document instance and is designed to provide formatting information.

6.5.2.4 Output Specification (OS). A finite set of style characteristics to convey formatting intent for interchange of publications coupled with a mechanism for binding the style characteristics to logical elements in an SGML document type declaration. The OS uses the syntax of an SGML document type declaration.

6.5.2.5 Standard Generalized Markup Language (SGML). SGML, as detailed in International Standard 8879, is a metalanguage that provides a coherent and unambiguous syntax for describing whatever a user chooses to identify within a document.

6.5.2.6 SGML declaration. A markup declaration that specifies the character set, concrete syntax, optional features, and capacity requirements of a document's markup. It applies to all of the SGML entities of a document.

6.6 SUBJECT TERM (KEY WORD) LISTING. The following subject terms (key words) are applicable:

Publishing, Electronic
Standard Generalized Markup Language
Tagging
Document Type Definition (DTD)
Formatting Output Specification Instance (FOSI)
Output Specification (OS)

6.7 CHANGES FROM PREVIOUS ISSUE. Marginal notations are not used in this revision to identify changes with respect to the previous issue due to the extent of the changes. As part of the CALS initiative to introduce the use of digital technology into the process of reviewing and coordinating standards, this revision of the standard has been reformatted for improved readability as both a paper and an electronic document. The body text of this document uses the same font as the previous revision, but sightly enlarged to give an improved on-screen appearance when displayed by a computer. The tables and figures now use a sans-serif font for a cleaner appearance and to be distinguished easily from the body text. Computer code entries, values, and listings are shown in a non-proportional typewriter font so that they may be identified easily and with minimal confusion.

APPENDIX A

RECOMMENDED CONTRACTING OFFICER RESPONSES

A.1 SCOPE

This appendix is designed to assist the users of this specification in placing the optional requirements of this specification on contract. It provides generic guidance and explanations to assist in determining the appropriate decisions to make. This appendix is not a mandatory part of this specification. The information contained herein is intended for guidance only.

APPENDIX A

A.2 APPLICABLE DOCUMENTS

THIS SECTION IS NOT APPLICABLE TO THIS SPECIFICATION.

A.3 SUGGESTED GUIDANCE

A.3.1 PACKAGING REQUIREMENTS. Packaging requirements for delivery of any DTDs or FOSIs developed to this specification shall be as determined in the contract or other form of agreement. MIL-STD-1840 provides additional guidance on procedures to identify data that is being packaged for delivery.

A.3.2 SGML REUSE. The DoD recommends the reuse of existing DTDs, FOSIs, and tags to the maximum extent possible. To this end, the DoD has established an SGML tagset registry and library. Participation in this program is voluntary. Use of existing objects will in the long run save the DoD time, money, and improve data interchangeability and standardization.

Service department and agency procedures on the use of the CALS SGML Registry (CSR) and CALS SGML Library (CSL) should be followed. Procedures for submission of SGML objects to the CSL or use of objects currently available are in the CSL Construct and Object Submission Procedures (see 3.3.2).

A.3.3 PRESENTATION CHARACTERISTIC VALUES. Where the declared value of a presentation characteristic in the OS is derived from a list or the implied value generated by the system is not adequate, the contract or other form of agreement should specify the desired declared value for that characteristic.

A.3.4 ADDITIONAL NOTATION DECLARATIONS. For most uses, the notation declarations contained in detail specifications will be adequate. There may be cases where additional notation declarations are required.

When a notation declaration not found in a detail specification is required, the contract should specify the following:

a. A formal public identifier.

b. A system identifier.

The system identifier should provide sufficient information to locate the application that supports the notation.

A.3.5 DEVELOPMENT OF PROGRAM SPECIFIC DTDS OR FOSIS. Normally, the DoD desires to use existing DTDs and FOSIs to prepare and deliver technical or administrative data. This allows for easier exchange of similar data to dissimilar users. If there are DTDs and FOSIs existing for the types of data being acquired, they should be used. The public identifiers for the applicable DTDs or FOSIs should be included in the Request For Proposal (RFP). Information on existing DTDs and FOSIs is available from the CSL. Procedures for accessing the library may be obtained from the CALS Digital Standards Office, DISA Center for Standards, Code JIEO/JEBEB, 10701 Parkridge Blvd, Reston, VA 20191–4357.

A new DTD or FOSI should be prepared only if:

a. A DTD or FOSI does not exist.

b. The responsible government agency is not preparing the needed DTD or FOSI.

APPENDIX A

The RFP and contract should require development and delivery in accordance with the requirements of MIL-PRF-28001. Service or agency requirements may differ, and the program data manager should verify project requirements with the appropriate service/ agency authority.

A.3.6 SUBDOC. SUBDOC is an SGML optional feature that has been referenced in this specification. SUBDOC provides the capability to include SGML subdocument entities in an SGML document. The SUBDOC feature is not supported by some SGML applications. The contract may prohibit the use of SUBDOC to avoid potential interchange problems.

A.3.7 CONFORMANCE. Paragraph 4.1 establishes the requirement to parse SGML documents prepared to this specification with a validating parser. No single parser records and displays all potential errors. The DoD has not certified any parsers. To ensure maximum compliance with the requirements of ISO 8879 and this specification, contracting and acquisition managers should establish additional conformance inspections on each SGML document.

A.3.8 FORMATTING OF MATHEMATICAL ELEMENTS. The MIL-PRF-28001 OS does not support the specification of formatting characteristics for mathematical elements. When creating an SGML-tagged source file tagged for a specific document or contract, mathematical elements must be handled in one of four ways:

a. Simple inline mathematical notation can be formatted as regular text using special characters, superscripts, and subscripts.
b. Some complex mathematical notation can be generated with an illustration program and included or referenced as graphics in the SGML-tagged source file.
c. Other complex mathematical notation can be formatted using a specialized mathematical formatting tool where the composed notation is provided in the form of graphics files which are specified in the SGML-tagged file. It is recommended that the source material for these graphics files be provided with the SGML-tagged source file to permit modification of the notation at a later date.
d. SGML tagging can be employed for marking up simple and complex mathematics. However, the current state of the Output Specification does not support the association of complex mathematical formatting specifications to arbitrary SGML declaration sets. This option must be evaluated in terms of the practical availability of composition systems that will be able to format such SGML-tagged mathematics. It should not be assumed that such systems necessarily are or will be widely available.

For additional information on the tagging of mathematical notation, see MIL-HDBK-28001.

A.3.9 PARTIAL DOCUMENTS. Partial document delivery is used to transmit SGML source data either as an interim deliverable or as an update package for a document that has been previously delivered. Its purpose is to minimize the retransmittal of unchanged data, or to indicate data that is incomplete. Partial document delivery is not intended to address the issues of page integrity or fidelity, nor is it intended to address change

pages. However, delivery in this form does not preclude maintaining page integrity for change page delivery. The intent of this methodology is to allow the transmission of portions of a source document so the receiving system can identify the location of the information in the original document and perform the appropriate add, delete, or replace operation. The manner in which this is accomplished and the effect of the change on composition is up to the receiving system and should reflect the requirements called out in the controlling specifications. For additional information on partial document delivery, see MIL-HDBK-28001.

APPENDIX B

OUTPUT SPECIFICATION

B.1 INTRODUCTION

B.1.1 SCOPE. This appendix describes a method for interchanging formatting requirements for military technical documents whose source files are tagged according to DTDs developed in accordance with this specification. Adherence to the rules described in this appendix allows for divergent receiving processing systems to unambiguously interpret the style and formatting intent of the sending system. This is accomplished by combining the tagged source file with the appropriate FOSI. The resulting publication will preserve the information content of the original document while maintaining a similar presentation.

This appendix is not a mandatory part of the specification. However, when this appendix is cited in the contract, the information cited herein is intended for compliance. The following is a partial list of functionality that is not provided for in this version of the Output Specification (OS):

a. Formatting mathematical elements.
b. Supporting all functions necessary for producing change packages.
c. Supporting all possible security classification markings.
d. Supporting all semantics of interactive electronic presentation.

B.1.2 STATEMENT OF PURPOSE AND PREMISES. This specification is designed for use with all military specifications for technical documents. The military functional specification, such as MIL-STD-38784, determines the actual requirements. A specific DTD interprets the content and structural requirements of a particular functional specification, and a specific FOSI interprets the style and formatting requirements of the functional specification. The designer of the DTD and FOSI is responsible for assuring that they convey a consistent, unambiguous, and complete description of the pertinent areas of logical tagging and output presentation from the military specification. It is a requirement of this specification that FOSIs be rigorous. If a particular contract calls for an exception or variant interpretation of a functional specification, then an unambiguous FOSI must be created. This appendix describes the rules for creating all FOSIs to be included in or delivered in accordance with this specification, as well as the interchange format to be used. Throughout this document the terms “Output Specification” and “OS” refer to this appendix, and the terms “Formatting Output Specification Instance” and “FOSI” refer to a particular application of the rules and methods set forth in this appendix in accordance with a military functional specification, such as MIL-STD-38784.

B.1.2.1 Media. The physical form on which information will be output, paper or electronic display screen, is known as the media. Although mainly concerned with paper media, the OS does allow some electronic presentation capabilities. Throughout this appendix, the term “page” can be interpreted as a page of paper, an electronic screen, or the area inside the frame of a window on an electronic display.

B.1.2.2 Design goals. The overall goal of this output specification is to allow for the interchange of style and formatting information of technical documents between all types

of publishing systems. This includes current batch and print screen systems, as well as future systems incorporating newer technology. This is accomplished by the interchange of formatting style information, using the semantics described, to be used as input to the formatting system.

B.1.2.3 Page integrity and page fidelity. Page integrity in this specification is defined to mean the ability to preserve the same information on each page in a document as it is exchanged between systems. This does not mean that the information will be presented exactly the same way, but only that it will appear between the same page boundaries. Page fidelity is defined to mean the ability to preserve the exact presentation characteristics in addition to the same information on pages exchanged between systems. Preserving page fidelity between systems is not technically feasible with current technology. Preserving page integrity is possible to some degree with today's technology, but has the following consequences:

a. The pages may look different.
b. Additional cost may be associated with the effort to ensure that the pages are identical in content and presented in a readable fashion.
c. In order to preserve page integrity as much as possible and still adhere to the required FOSI, the author of the FOSI needs to be sure to include enough flexibility in the specification of characteristic values to allow pages to appear differently.
d. Complete page integrity may not be compatible with arbitrary SGML documents or certain FOSI specifications.

While complete page integrity is not supported by this specification, it is not precluded by it. Page integrity requirements should be carefully reviewed in the context of applicability, usability, and cost.

B.1.2.4 Machine processability. FOSIs prepared in accordance with the OS DTD in B.5 are machine parseable. Machine parseability is defined here to include the ability for a machine to automatically verify that a FOSI contains all the required characteristic values and is presented in the correct syntax.

B.1.3 ORGANIZATION OF THIS APPENDIX. The remainder of this appendix contains the following major sections:

a. Section B.2. Applicable documents.
b. Section B.3. Output Specification (OS) concepts. This section explains concepts that are important for a complete understanding of the OS.
c. Section B.4. Key to characteristics. This section describes in detail the types of characteristics and the rules for applying them, including ranges of values when generating a FOSI.
d. Section B.5. Output Specification (OS) DTD.
e. Section B.6. Glossary. This section contains the definitions of all significant terms used in the Output Specification.

APPENDIX B

B.2 APPLICABLE DOCUMENTS

B.2.1 GOVERNMENT DOCUMENTS.

B.2.1.1 Government standards.

STANDARDS

FEDERAL

FIPS PUB 152 - Standard Generalized Markup Language (SGML)

(Copies of the Federal Information Processing Standards (FIPS) are available to Department of Defense activities from the Standardization Document Order Desk, 700 Robbins Avenue, Building 4D, Philadelphia, PA 19111-5094. Others must request copies of FIPS from the National Technical Information Service, 5285 Port Royal Road, Springfield, VA 22161-2171.)

B.2.1.2 Government handbooks.

HANDBOOKS

DEPARTMENT OF DEFENSE

MIL-HDBK-28001 - Application of MIL-PRF-28001 Using Standard Generalized Markup Language (SGML)

(Unless otherwise indicated, copies of the above specifications, standards, and handbooks are available from the Standardization Document Order Desk, 700 Robbins Avenue, Building 4D, Philadelphia, PA 19111-5094.)

B.2.2 NON-GOVERNMENT PUBLICATIONS. The following document(s) form a part of this document to the extent specified herein. Unless otherwise specified, the issues of the documents which are DoD adopted are those listed in the issue of the DoDISS cited in the solicitation. Unless otherwise specified, the issues of documents not listed in the DoDISS are the issues of the documents cited in the solicitation (see 6.2).

ISO 8879 - Information Processing — Text and office systems — Standard Generalized Markup Language (SGML)

ISO 9541 - Information Technology — Font Information Interchange Part 1 — Architecture — 1st Edition Part 2 — Interchange Format — 1st Edition Part 3 — Glyph Shape Representation

ISO 9594 - Information Technology — Open Systems Interconnection — the Directory

(Application for copies should be addressed to the Standardization Document Order Desk, 700 Robbins Ave., Building 4D, Philadelphia, PA 19111-5094, for issue to DoD

activities only. All other requestors must obtain documents from the American National Standards Institute, 11 West 42nd Street, 13th Floor, New York, NY 10036.)

B.2.3 ORDER OF PRECEDENCE. In the event of a conflict between the text of this document and the references cited herein, the text of this document takes precedence. Nothing in this document, however, supersedes applicable laws and regulations unless a specific exemption has been obtained.

B.3 OUTPUT SPECIFICATION (OS) CONCEPTS

B.3.1 BACKGROUND AND BASIC CONCEPTS.

B.3.1.1 Overview. The SGML standard (ISO 8879:1986) does not define a formatting language. For SGML-encoded information to be composed, application semantics (for example, style sheets in the simplest case) must be associated with the various markup constructs used to encode the given information.

The most obvious need for associating semantics is the need to attach formatting style to the various parts of the information so that the information can be presented (online or on paper) to the user as a readable document. If arbitrary documents need to be interchanged among arbitrary systems in such a way that comparable printed results are produced, then either the software used must be hardwired to a given, fixed DTD that has style information for that DTD built into the software, or a separate style sheet, must be interchanged. If a commonly recognized output specification is included in the interchange package, then the information is guaranteed to be interchanged successfully and there is a level of confidence that the document can be presented by the receiving system as was intended. This appendix defines an Output Specification (OS) that describes a class of style sheets called Formatting Output Specification Instances (FOSIs) that can be used for CALS-compliant interchange of formatting style.

Most solutions to the problem of how to format an SGML document make use of one or more basic components or techniques. The process can be outlined as follows:

a. The possible transformation to another (SGML) document on which the remainder of the process continues.
b. The use of some “location model” to address (that is, point) into the SGML document to indicate the object(s) to which formatting characteristics are about to be associated.
c. The association of some set of formatting characteristics with the addressed object(s).
d. The definition of some result object structure (that is, the “target” structural model for a composed document—for example, the page layout), and perhaps the specification of rules on how the document content gets “filled into” this structure.
e. A composition process that takes the input document and the various “semantic attachment specifications”, and maps them into the result object structure to create the formatted document.

The OS does not include a transformation step (see B.3.1.7 on the logical input and output stream structure). For its location model, the OS defines the concept of an “element-in-context”. The OS defines a set of formatting characteristics and groups them into a set of categories. To simplify and increase the power of the specification, the OS defines a method of inheritance and defaulting of characteristics. The OS allows for the definition of a range of page layouts and provides some additional control over how to fill the content from the source document into the output stream (including counters, string

variables, and floating constructs). The OS avoids defining the composition process; rather it defines the specification of style attachment in such a way that various composition processes and tools could be used.

B.3.1.2 A Formatting Output Specification Instance (FOSI). The OS is defined in terms of a DTD. B.5 consists of this DTD. The OS defines an overall style sheet format for the definition of simple page layout, association of various style characteristics to elements, simple table styles, graphic (figure) styles, and simple footnote styles.

An SGML document instance that conforms to the OS document type is called a FOSI. A FOSI is:

a. An SGML document instance in the OS doctype.
b. A style sheet combining all the elements' style information as well as some page layout information.
c. A style sheet for a class of documents, not necessarily only for a single document.

There can be as many FOSIs for a single source DTD as there are different styles for documents using that DTD. Generally, a given FOSI would be used for only one DTD (or for a class of quite similar DTDs).

B.3.1.3 Element-in-context (e-i-c). FOSIs attach style to elements-in-context (e-i-cs). That is, context and occurrence conditions can be specified in addition to the element type name, Generic Identifier (GI), so that element instances of the same name but in different contexts can be given different style. "Context" here refers to the context in the SGML input structure (see B.3.1.7). When more than one specification holds for a given element, the most specific one according to B.3.6.2.2 takes effect. The e-i-c's use of the GI, context, and occurrence attributes makes up the OS's location model. The contents of each e-i-c contain a component—called the characteristic list or "charlist"—that specifies the set of formatting characteristics to associate with elements that match this e-i-c.

The e-i-c concept also applies to two other constructs besides actual SGML elements declared in the source DTD; pseudo-elements (see B.3.6.6), and processing instructions (see B.3.6.7).

B.3.1.4 Categories and characteristics. Any definition of an OS defines a set of formatting characteristics that can be associated with the various objects in the document. The OS includes characteristics such as font size, font weight, left indent, right indent, quadding (horizontal justification), and so forth. For convenience and ease of inheritance, the OS groups related characteristics into categories. For example, the font category includes the following individual characteristics (and some sample values): style (serif, sanserif, monospaced), size, posture (upright, italic), weight (regular, bold).

The categories (groups of characteristics) as described in B.4 are represented in the OS DTD as elements. Individual characteristics are represented as attributes on those elements. The content models for elements may represent the interaction between categories.

A characteristic is a specification of a particular formatting property that is to be associated with a specified element instance by the composition process. Characteristics have allowable

ranges of values. In a FOSI, every characteristic has a value that specifies in some way how the processing system should treat the associated content. Characteristics are descriptive specifications of the desired format of a document, rather than commands that tell a formatting system what to do. In particular, this OS is not a formatting language, nor does it presume to direct how a formatting system should behave in order to accomplish the desired result.

It is important to understand that the terms “characteristic” and “category” refer to concepts that are represented in the OS DTD and the resulting FOSIs by the SGML constructs of attributes and elements. Whereas all characteristics are represented by an attribute on some element, not all attributes of all elements in the OS DTD represent characteristics. In particular, the “inherit” attributes present on some of the elements in the style description section of the OS are not characteristics as they do not (directly) affect formatting. These attributes are used by the inheritance mechanism to determine the resolved values of the characteristics.

Though an element instance may have a (possibly inherited or defaulted) resolved value for a variety of characteristics, not all characteristics are relevant to every element instance. For example, the “prespace” category’s characteristics have no formatting effect on an “inline” e-i-c such as might be used for an “emphasis” element. These issues are addressed in the text of this OS specification. See B.3.9 for further information on inline versus block structures.

B.3.1.5 Environments and characteristic subsets. The effective settings of the collection of all characteristics forms the formatting “environment” at any given time. The OS allows for a complete collection of settings to be designated a named environment that can be referenced from an e-i-c. An environment is referenced in such a way as to present a set of defaults that will take effect if the e-i-c in question does not make an explicit setting, hence such environments are often known as “default environments.” Most default environments have a name associated with them when they are defined—these are known as named environments. The single unnamed environment is the document description (docdesc) environment—the environment of last resort (short of output system provided defaults)—that is defined in its special place in the FOSI. While an environment may be defined by specifying only a few characteristics explicitly, it then always references another environment explicitly or will implicitly default to the document description environment so that a resolved environment is effectively always a complete set of assignments for the full set of characteristics. In particular, it would never make sense to refer to two environments in a given e-i-c, since the second would completely replace the first. Therefore the utility of environments is restricted to providing generally only a small set of overall defaults. In documents where the overall defaults tend to be the same throughout most or all parts of the document, it is common to have no named environments but just to use the docdesc to provide defaults.

Characteristic subsets (charsubsets) work more like “macros” or “packages” of characteristic settings. A charsubset consists of any subset of characteristic settings that could be defined in a charlist—they do not define a complete environment. Hence, charsubsets can be

combined to form other charsubsets and charlist. The resulting charlist (see B.3.7 and B.3.6.5.2.4) will use the default environment to provide settings for characteristics left unspecified only after all settings provided by all charsubsets and any explicit settings have been considered.

B.3.1.6 Introduction to inheritance and defaulting. There are many characteristics that can potentially be specified in a charlist. In practice, however, most e-i-cs omit explicit specifications for many of the characteristics. The inheritance and defaulting conventions provide mechanisms for determining values for unspecified characteristics in an e-i-c's charlist.

Inheritance and defaulting are two distinct cases that have specific meaning in the definition of the OS, and the OS defines which is applicable under what circumstances. For each category that is allowed in an e-i-c's charlist, the OS defines it as "inheritable" or not based on the presence of the "inherit" attribute on the element defining that category. Defaulting is applicable for non-inheritable categories in a charlist. The precise definition of inheritance and defaulting appears in B.3.7. Briefly, inheritance is a way for unspecified characteristics in inheritable categories to have their value derived from that of their parent's formatting environment. Defaulting refers to the process whereby unspecified characteristics (both in inheritable categories that choose to default rather than inherit and in uninheritable categories) derive their value from a named environment (or from the docdesc).

B.3.1.7 Logical input and output stream structure. While the OS does not include an explicit transformation step in the sense described in B.3.1.1, the logical structure of the input and output stream differ in some important ways. Both structures can be represented as hierarchies of nested substructures or "trees". A structure that "is immediately contained within" another structure is referred to as its "child" and the containing structure is known as the "parent". By extension, any structure that contains (at any number of levels lower) another structure is identified as an "ancestor" and the contained structure is a "descendant".

In the SGML source input, the ancestor/descendant relationships are generally defined by the SGML standard. (In fact, the OS-defined input tree is not quite equivalent to the SGML tree, since generated text such as possible with Usetext does allow the introduction into the input tree of structures such as pseudo-elements.) Context determination (for e-i-cs) is done according to the ancestor/descendant relationships in the input tree. For example, a section title's parent is the section whose parent may be a chapter whose parent may be the body element whose parent may be the document element. The document element is generally the "root" of the tree (the ultimate ancestor) and has no parent unless it is the document element of a SUBDOC entity that is referenced from another document. For the purposes of the OS-defined input tree (and especially for the purposes of defining e-i-cs), the parent of the SUBDOC's document element is deemed to be the referencing document's element that contains the reference to the SUBDOC entity. This could be either as part of its content or as the value of an ENTITY attribute.

Pseudo-elements may get introduced into the input tree via occurrences of the Savetext and Usetext categories. Pseudo-elements are also matched against the e-i-cs in a FOSI, and the context of any pseudo-element is determined (within the input tree at Usetext time) in the

same fashion as is the context for an SGML element. Processing Instructions (PIs) occur at given points within the input tree (that is, they too have a certain context in the input tree for the purposes of e-i-c matching), but they do not contribute to the structure of the input tree themselves. As such, they shall not contain (that is, be the ancestor of) any other structures in the input tree.

While the input tree determines context for the purposes of determining e-i-cs, the output tree structure determines the ancestor/descendant relationships for the inheritance mechanism. The result of each e-i-c in the output tree determines a set of characteristic settings that can be inherited by descendants in the output tree. While the ancestor/descendant relationships in the output tree for the most part parallel those in the input tree with respect to elements and pseudo-elements, there is a significant difference with respect to the special OS-recognized PIs. The OS-recognized paired processing instructions, while they do not contribute structure to the input tree, do contribute structure to the output tree. The characteristic changes made within (and scoped in the output tree by) an OS-recognized PI pair do form a structure in the output tree whose characteristic settings can be inherited by its descendants in the output tree.

B.3.1.8 Page layout. One part of the OS describes the various page layouts that will be used by the composition system to format pages. A wide range of page layouts, but not all possible page layouts, can be described by the OS. Capabilities such as multiple columns, header and footer areas, footnote areas, areas for floating objects (such as figures), different recto and verso and division-opening pages are all available.

B.3.1.9 Complex style issues (simple transformations). Facilities exist to do “simple transformations” even though the OS does not include a complete transformation step as mentioned in B.3.1.1. There is, however, no current capability to transform the current SGML document into another one.

The OS includes some capabilities for managing numeric variables (counters) and doing simple manipulation of string variables (also knows as textids) that allow information from one point in the input document to be captured and placed in another location in the output stream. (See B.4.1.3 and B.4.4.25 for information on counters; see B.3.4.7, B.4.4.29, and B.4.4.30 for information on string variables.)

To cover the case where document content should not be passed through to the output stream—either because it will be picked up for placement elsewhere or because it should not be displayed at all in this output stream—the OS includes a “suppress” category (B.4.4.20) in the Style description.

Especially for print composition, certain constructs are naturally meant to be placed in the output stream in a position determined by the composition process rather than by the order of the input. This includes constructs such as footnotes as well as other objects that have been specified as being allowed to “float” to some formatter-determined location. It is common to allow certain tables, graphics, figures, and other illustrations to float to a particular place on a given page or to some other page (see B.3.13).

B.3.2 OVERVIEW STRUCTURE OF THE OS. The DTD contained in this appendix is a rigorous formalization of the functionality presented in B.4 “Keys to characteristics”. The subsections of the “Keys to Characteristics” section are reflected in the structure of the DTD, which is divided into eight major functional areas. The resource description, security description, and page description areas establish the information for page layout and other document-wide rules. The next four sections of the DTD—style description, table description, graphics description, and footnote description—provide the means by which elements in the source document are identified and define the processing that is to occur for each element. All elements whose specification occurs in the general style description (styldesc), table description (tabdesc), or graphics description (grphdesc) are placed into the flowing text area of the page. Elements whose description occurs in the footnote description (ftndesc) are placed into the footnote area. The last section, the change description (chngdesc) provides specifications for change page processing.

B.3.2.1 Resource description (rsrcdesc). The resource description gives document-wide hyphenation rules, establishes strings, and provides descriptions of character fills, counters, and floating locations that will be used throughout the FOSI.

B.3.2.2 Security description (secdesc). The security description sets up the guidelines for handling variable security markings on a page.

B.3.2.3 Page description (pagedesc). The layout geometry of pages is defined in the page description area. Within this section, page sets (pageset) are established to reflect the different styles of pages that may need to be used within the document. Each page set describes a right-hand (rectopg), left-hand (versopg) page, and a page to be used when the formatter generates a blank page (blankpg). The header and footer may be redefined when the formatter generates a recto page with a blank back (rectobb), or a verso page with a blank front (versobf).

B.3.2.4 Style description (styldesc). The style description provides the means for identifying e-i-cs in the source and the characteristics to be applied to them as they are processed. In addition, attribute values from the source document may be identified and used to affect the formatting process.

B.3.2.5 Table description (tabdesc). Similar to the style description, elements that are processed as tables are identified and described through the table description.

B.3.2.6 Graphics description (grphdesc). Graphic elements are identified and described through the graphics description.

B.3.2.7 Footnote description (ftndesc). Footnote elements are defined in the footnote description area.

B.3.2.8 Change description (chngdesc). The change elements for the composition of looseleaf change packages are described in the change description area.

B.3.3 CHARACTERISTICS. There are two basic types of characteristics: Composition and Pagination.

B.3.3.1 Composition characteristics. Composition characteristics define how a particular element, when it is encountered, is to be treated. Composition characteristics are grouped into the following functional areas:

a. Style characteristics, which generally apply to all elements.
b. Table characteristics, which apply only to elements presented as a table.
c. Graphic characteristics, which apply only to elements presented as a graphic.
d. Footnote characteristics, which apply only to elements presented as a footnote.

It is convenient to think of each element as having a formatting environment associated with it. An element's formatting environment is the complete set of characteristics and their values that are in effect during the scope of that element. The start-tag of the element opens or begins the scope of the element, and the end-tag of the element closes or terminates the scope and causes the scope of the parent element to be resumed. During the scope of any element, the start-tag of a child element will define a new scope with a new and potentially different formatting environment that will remain in effect until the end-tag of that child element.

Anything generated by an e-i-c (as specified by categories such as usetext, puttext, putgraph, ruling, chgmark, textbrk, presp, postsp, border, boxing) is considered to be generated within the scope of that e-i-c and is therefore affected by the formatting environment associated with that e-i-c. The fact that an element instance has no content (that is, there is nothing in the document between the element's start and end tag) does not change the processing of that element instance's e-i-c (except via the standard attribute specification evaluation process whereby an e-i-c could test the #CONTENT of the element instance). Elements with EMPTY declared content in the source DTD and element instances with a specified CONREF attribute (element instances which have no end-tag in the normalized document) follow similar rules. Even though they can contain no document instance content, they form a scope for any generated material, and their e-i-cs are processed in full and may contribute material (such as text, graphics, spacing, or borders) to the output stream.

In a FOSI, the specification of categories of characteristics in an e-i-c is contained within a construct known as the characteristic list (charlist). The explicitly specified charlist may be augmented in the e-i-c specification with references to predefined named environments, predefined named characteristic subsets (charsubsets), and further characteristic subsets that may be conditionally called into force depending on the values of certain attributes on the source (document) element instance. The process for combining the specified charlist with these various augmentations is defined in this appendix, and the result is known as the merged charlist. Finally, the inheritance and defaulting process described in this appendix is applied to produce what is called the resolved charlist, which defines the formatting environment that will be in force for the scope of each element instance that uses this e-i-c.

B.3.3.2 Pagination characteristics and layout areas. Pagination characteristics define the layout of a page, independent of the elements occurring on that page. They define the following type of information:

a. The physical aspects of pages (the page model), including size, margins, and areas for placement of text.

b. How header, footer, and change mark areas appear on every page.

c. Placement rules for floating elements.

Pagination characteristics differ from composition characteristics. Composition characteristics are attached to elements. Pagination characteristics are applied to the areas on a page within which the output resulting from composing the elements is placed. These areas are referred to as “layout areas.” Examples of Layout Areas are “column area” and “header area.” Other than this basic difference, the application of characteristics is similar. A page model defines the structure of a page by specifying which Layout Areas are permissible, how these Layout Areas relate to each other, and what characteristics may be attached to them.

B.3.4 VALUE TYPES. This section describes the eight types of values that characteristics may have. The actual values to be used are specified in a FOSI. Each kind of value is named and then explained. Allowable value ranges are included in this specification where appropriate. Where value ranges are specified in this appendix, a particular FOSI may only specify values from these ranges.

B.3.4.1 Units of measurement. Points are the basic unit of measurement used throughout this specification. A point is defined as being equal to 1/72 of an inch with a tolerance of +/- 0.4 percent. It is recognized that most typesetting systems use a more precise measurement for points, and that such systems may need to convert parameters given in points in a FOSI to another unit of measurement for use within their own system. Alternative units of measurement are allowed. The syntax to be used for expressing measurement in the allowable types of units is described in B.3.4.2. Points shall be used as the default unit of measurement for the interchange of FOSIs.

The use of points as the basic unit of measurement does not preclude any finer levels of precision. Finer levels of precision may be expressed as either tenths of a point or hundredths of a point.

B.3.4.2 Size/distance. This type of value is used generally for specifying characteristics that express measurement. The syntax for describing a Size/Distance value is number unit, for example “6pt” means 6 points. Numbers may be positive or negative, and may express precision in tenths or hundredths with the use of decimal points, for example, “6.4pt” means 6 and 4/10ths of a point. Each unit is expressed as a two letter abbreviation. Combinations of units are allowed, but must be completely specified; for example, to specify 5 picas and 3 points, the correct syntax is “5pi 3pt” (the space separating the two individual unit specifications is optional). Allowable expressions of units are:

pi	picas	pt	points	in	inches
mm	millimeters	cm	centimeters	em	em space

A size/distance of 1 em is a measurement equal to the current font size and is therefore a relative rather than an absolute specification. (For example, for a font size of 10pt, a specification of 1.2 em evaluates to 12pt.) The evaluation of a size distance specification using units of ems (that is, the conversion from ems to an absolute value) happens in the charlist in which the specification occurs (explicitly or via defaulting). Only the absolute value is available to be inherited. However, when a size/distance specification using units of ems which is contained in a named environment or docdesc is used as the source for a default, the (relative) specification is filled into the charlist before it is converted into an absolute value.

B.3.4.3 Integer. Integers are whole numbers that are either positive, zero, or negative. A "-" prefix to the magnitude designates a negative integer. Examples of correct values are "2", "14", and "-4". Examples of incorrect values are "3.324" and "6 3/4". Some characteristics are restricted to the use of positive Integer values. These are described where the individual characteristic is defined. Percentages are specified by a positive integer (where 100 indicates 100%).

B.3.4.4 ID and IDREF. An ID is used to uniquely identify a characteristic or set of characteristics so that it can be referenced by an IDREF. ID and IDREF are used as in ISO 8879.

B.3.4.5 Pointer. Pointers are used to specify that information is contained in an external file. Pointers have the attribute type of "entity" and thus require an entity declaration in the FOSI declaration subset to identify the external file.

B.3.4.6 Finite list. This type of value is used where a finite and fixed set of choices is to be made for a particular characteristic. That is, wherever a finite list is specified, all legal values are explicitly contained in the list. For example, there is a well-defined short list of possible values for the Quadding characteristic (Flush Left, or Justified).

B.3.4.7 String. A string is a literal sequence of characters. References to non-keyboard characters are made with an entity reference using the "&" delimiter and the name of the entity, followed by a semicolon, for example "†". To specify an "&" character when it is followed by a name character, use an entity reference for the ampersand (IV&V) which resolves to IV&V. A string differs from a finite list in that the values are not known in advance. The syntax to which a given string must conform depends on its context in a FOSI.

B.3.4.8 Toggle. Zero and non-zero are used for toggling purposes. Zero is "0", "00", "000", and so on (any number of zero characters up to limits imposed by the SGML declaration in force) and non-zero is any other integer, for example, "1" or "8" or "23". Zero is defined as 0, false, no, and off. Non-zero is defined as 1, true, yes, and on.

B.3.4.9 Indent syntax. The indent category's Left Indent (leftind), Right Indent (rightind), and First Line (firstln) characteristics take a Size/Distance as values. However, in the case of these three characteristics, their possible values include some special extended syntax.

The syntax for indents allows for specification in two ways:

a. With respect to the boundary of the current Layout Area, for example, “+2pi” or “–2pi”. A Size/Distance value appearing alone specifies a distance from the margin of the current Layout Area. For a left indent specification, a “+” indicates an indent to the right and a “–” indicates an indent to the left (a “hanging indent”). For a right indent specification, a “+” indicates an indent to the left and a “–” indicates an indent to the right. If no “+” or “–” appear, a “+” is assumed.

b. With respect to the text margin determined by the parent element’s indent, for example, “@+2pi” or “@–2pi”. The delimiter “@” can be used to specify that the indent is relative to the text margin established by the element’s parent, including any indenting that may have been applied. For example, if a para element were indented two picas from the left margin of the Layout Area and the specification for its child warning is “@+2pi”, the warning will be indented 2 picas from its parent, or 4 picas from the left margin of the current Layout Area.

With regard to inheritance, when the “@” syntax is used in the specification of an e-i-c, the “@” is expanded for the e-i-c instance where it is specified, and what is inheritable is the resulting value. For example, if the current left margin is 5 picas and an element’s indent specification includes leftind="@+2pi" then this element’s left margin will be 7 picas. If this element’s child’s indent specification is <indent inherit="1">, then the child should inherit the parent’s indent, therefore having an indent of 7 picas, rather than inherit the parent’s indent specification of “@+2pi” (where now the “@” refers to the parent’s 7 pica indent), which would give it a 9 pica indent.

The syntax for First Line also permits an “*” to be used as the first character of the First Line value to refer to the (resolved) value that the Left Indent has for this e-i-c instance. That is, to get the effect of having a First Line indent equal to the Left Indent, (which would give a “block indented” paragraph where all lines have the same left indent) one would set firstln="*". Note that to specify a non-zero First Line indent that is relative to the Left Indent, the value of First Line might be “*+15pi”. When the Left Indent is specified and the First Line indent is not, the value for First Line indent is determined by defaulting and inheritance rules. If this value is “*”, it is replaced with the value of Left Indent of the current e-i-c.

The syntax for First Line and Left Indent also permits a “^” to be used as the first character of the indent value to specify that the indent is relative to the first line indent established by the element’s parent. For example, this allows the formatting of nested paragraphs in which the first line indent of each paragraph is relative to its parent’s first line indent.

The delimiter “*” can be used as the first character of the Right Indent value to specify that the Right Indent is to be relative to the Left Indent rather than relative to the right boundary of the text area (and that a positive offset will be to the right of the Left Indent). This allows the Right Indent to be specified in terms of the Left Indent and the width of the block of text. For example, in a layout area of 45 picas, a Left Indent specification of “2pi” and a Right Indent specification of “*+18pi” would produce text with a 2 pica left indent and a 25 pica right indent (and in a layout area of 30 picas, the

same specification would produce text with the same width even though the right indent would now be only 10 picas).

If the "@" or "*" or "^" syntax is used in the docdesc or a named environment, the (unexpanded) indent specification itself would be expanded at the time the environment is referenced by a given e-i-c. (Note: this differs from how these special syntax characters are interpreted when they are used directly in the specification of an e-i-c.) This way, for example, if one specifies a relative First Line indent of "*+10pt" in the docdesc, all paragraphs would, by default, have a 10pt indent regardless of the particular Left Indent of the paragraph. Once an e-i-c's indent is determined from an environment as discussed in the previous sentence, the "@" or "*" or "^" has been expanded and is gone; if an offspring of this e-i-c inherited its indent, it would inherit the resolved indent values of its parent.

B.3.4.10 Construction rule syntax. A Construction Rule is a special type of attribute value used in the Savetext and Usetext categories (see B.4.4.29 and B.4.4.30). For the construction of "string" variables, that may include the specification of the e-i-c's element content; the e-i-c's element's attribute's values; the resolution of the e-i-c's element's cross-reference attributes; literal text; references to saved text or counter values (string and counter variables); spacing and padding with a unit expression; and pseudo-elements (see B.3.6.6). Each of the preceding objects is specified within a construction rule in its own kind of entry with a particular syntax; entries within the construction rule are separated by a comma and optional spaces.

B.3.4.10.1 Referencing the e-i-c's element content. Element content in the Construction Rule is denoted by "#CONTENT". (If "#CONTENT" does not appear within a construction rule, no content is saved with this construction rule. When "#CONTENT" does appear, the entire content of the element, including any children it may have, is saved, so it can have an effect when the saved content is used.)

B.3.4.10.2 Referencing the e-i-c's element's attributes' value. In the Construction Rule, an attribute value of the current element is referenced by an entry of the form "#CONTENT(attribute-name)" in the construction rule which ordinarily expands to the value assigned to the attribute in question. However, this construct will evaluate to the empty string:

a. If attribute-name references an attribute that has been assigned no value in the source instance element.
b. If the named attribute is not a valid attribute for the current element.

If #CONTENT(attribute-name) references an attribute of declared value ENTITY or ENTITIES, #CONTENT(attribute-name) will evaluate to the string containing the entity name or names provided as the value of the attribute. In the case of ENTITIES, the #CONTENT(attribute-name) construct will return the string that is the concatenation (separated by a single blank) of each string that would be returned by each individual ENTITY listed in the ENTITIES value in the same order as the listed ENTITIES.

If #CONTENT(attribute-name) references an attribute whose declared value is ID, IDREF, or IDREFS, the attribute's value is treated as CDATA (character data or text) and the name(s) of the ID, IDREF, or IDREFS is returned as text.

B.3.4.10.3 Referencing the resolution of cross-reference attributes. To aid in the resolution of cross-references, the Construction Rule allows a construct of the form "#XREF(xref-attribute-name, textid-name)". This construct will evaluate to the empty string:

a. If xref-attribute-name is not the name of a valid attribute for the current element.
b. If it is an attribute of the current element whose declared value is neither IDREF nor IDREFS.
c. If it names an IDREF or IDREFS attribute that has been assigned no value in the source instance or has been assigned a value that does not match the value of a valid ID assignment in the same instance.

When the stated conditions are met, this construct will return the value of the string variable named by the textid-name in the context of the e-i-c associated with the element instance in the document instance that defined the ID that this IDREF matches. In the case of IDREFS, the #XREF construct will return the string that is the concatenation (separated by a single blank) of each string that would be returned by each individual IDREF listed in the IDREFS value in the same order as the listed IDREFS. Note that the SGML standard specifies that SUBDOC entities shall have separate name spaces for IDs; the OS has no special provisions for managing this issue; rather, it is the responsibility of the FOSI writer, document instance provider, and perhaps other non-FOSI processing tools to manage this issue effectively.

B.3.4.10.4 Literal text. Literal text, including the space character, is placed within back-slashes in the construction rule. References to non-keyboard characters are made via entity references, as in any string.

B.3.4.10.5 String and counter variables. References to the names of saved text or counters (FOSI variables) appear with no delimiters. References to variables that are declared in the resource description of the current FOSI module ("ospackage") will use the local value (that is, the value of the variable that is scoped by the FOSI module in which this Construction Rule appears) unless the variable name is prefixed by "global:" in the Construction Rule. All references to variables and named FOSI objects shall be processed in a case-insensitive manner.

Note that FOSI variables such as counters and previously saved text are expanded (that is, their values are determined) at the time they first appear in the Construction Rule. However, the context of any elements (that may have been saved within #CONTENT) or pseudo-elements that get saved or used in a construction rule, which will be used to determine which e-i-c to match in the FOSI as well as parentage for inheritance, shall be based on the context in which the saved text is finally used (via a Usetext).

The style of a counter may be specified to override the style assigned to it in the resource description by immediately following the counter name with "[style specification]", where

style specification is “AR” (Arabic), “RU” (Roman uppercase), “RL” (Roman lowercase), “AU” (Alpha uppercase), or “AL” (Alpha lowercase).

Ordinarily, when a savetext/usetext is associated with an e-i-c instance in the flowing text, the value of all time-dependent variables referenced in that conrule/source is the value at the time the e-i-c instances’s start tag (if placement is before) or end tag (if placement is after) is encountered. However, if a savetext textid variable’s value is set (via savetext) in the page resource (pageres) element, header, or footer of a page description, its “current value” at the time the e-i-c instance’s start/end tag is encountered depends on whether one considers the pageset processing to occur before or after the flowing text is collected for the current page.

The construction rule syntax supports modifiers of FI, TO, BO, FB, and TB on string (textid) variable references. When appended to a textid variable reference in a construction rule in the pageres, header, or footer, the FI modifier implies that the value of the variable should be the first value assigned to that textid on the page (or, if there are no assignments to this textid on the page, the value of this variable at the start of the page). The TO modifier implies that the value to be used is this variable’s value at the top (the very start) of the page. BO implies that the value to be used is this variable’s value at the bottom of the page. The qualifier FB implies that the value to be used is an ordered collection of all of the values of that variable on the page, including the values corresponding to the FI and BO qualifiers, and all values in between. The TB modifier implies that the value to be used is an ordered collection of all of the values of that variable on the page, including the values corresponding to the TO and BO qualifiers, and all values in between.

When a textid variable whose value is modified (by a reset or savetext) in the page resource element or (by a usetext) in the header or footer appears in a construction rule outside of the page description, one of the two modifiers [TO] or [BO] can be appended to it. The TO modifier implies that the value of the variable that should be used in this savetext/usetext is the value just prior to the associated pageset processing. The BO modifier implies that the value that should be used is that immediately following the associated pageset processing.

The FI, TO, BO, FB, and TB modifiers are not allowed on counter variables. However, counters share with string variables the same synchronization issues when a variable that is modified in one of the pagedesc or flowing text is referenced in the other. When a construction rule in the pageres, header, or footer makes a reference to a counter (as opposed to a textid variable) that is incremented or reset in the flowing text, the result shall be as if the counter reference had an implicit BO modifier. When a construction rule in an e-i-c being processed as part of the flowing text makes a reference to a counter that is incremented or reset in the pageres, header, or footer of the page description, the result shall be as if the counter reference had an implicit BO modifier. In either case, if a different behavior is desired, the counter should be assigned to a textid variable so that an explicit modifier may be used.

The results of any changes to a variable specified by savetexts or enumerates in the flowing text, whose value is also changed in any page resource, header, or footer, are undefined

unless all e-i-cs which specify such changes in the flowing text also force a new page (via the Start Page characteristic). In this case, changes (if placement is before) to the variable specified in the flowing text take effect before the pageset processing for the page which this e-i-c forces. For example, a chapter element might force a new recto page as well as reset the page counter. The effect of this reset (issued from an e-i-c in the flowing text) should occur before the enumerate in the pageres increments this counter.

B.3.4.10.6 Spacing specifications. Specification of spacing is a number followed by a unit expression (with an optional initial "@" or "*" to indicate a padding specification); such spacing is discarded by the composition system if the material is composed so that the spacing immediately precedes or follows a line break.

B.3.4.10.7 Pseudo-elements. References to pseudo-elements (see B.3.6.6) are indicated by giving the name of the pseudo-element delimited by (1) an initial "<" to indicate the start of the pseudo-element or (2) an initial "</" to indicate the end of the pseudo-element; and in both cases, the reference is also delimited by a final ">". Pseudo-elements provide a construct with which to reference an e-i-c and thereby attach formatting. The context of any elements (that may have been saved within #CONTENT) or pseudo-elements that get saved or used in a construction rule, which will be used to determine which e-i-c to match in the FOSI as well as parentage for inheritance, shall be based on the context in which the saved text is finally used (via a Usetext).

B.3.4.10.8 Examples. The following examples illustrate many of these constructs.

To express "CHAPTER 1", where the value of the chapter counter (chapct) is "1":

```
<savetext textid="chaptstr" conrule="\CHAPTER \,chapct">
```

Or alternately:

```
<savetext textid="chaptstr" conrule="\CHAPTER \,chapct[RU]">
```

Or to express the same title with two em spaces at the end:

```
<savetext textid="chaptstr" conrule="\CHAPTER \,chapct,2em">
```

To express a paragraph number with the chapter counter (chapct), a hyphen, and the paragraph counter (para0ct):

```
<savetext textid="chaptstr" conrule="chapct,\-\,para0ct">
```

The following FOSI fragment (which assumes elements "chapter" and "xref" as defined in the Example DTD) will cause the "chapter" e-i-c to save a string named "crossref" that contains the word "Chapter " followed by the current chapter counter so that any occurrences of the element "xref" whose xrefid attribute names the value of the ID of the given chapter element instance will print out the appropriate cross-reference text.

```
<e-i-c gi="chapter">
  <charlist>
    <savetext textid="crossref"
              conrule="\Chapter \, chapct">
```

```
<e-i-c gi="xref">
  <charlist>
    <usetext source="#XREF(xrefid,crossref)">
```

An initial “@” or “*” preceding a unit expression indicates that the contents of this conrule up to the point of this expression should be padded on the right with space until the width of this contents is the given amount. In the case of an “@”, if the content’s width already exceeds this amount, no padding is added. In the case of an “*”, if the content’s width already exceeds this amount, a new line is forced and the new line is padded out to the given amount. Therefore, to express a sequential list item number (seqlstct) padded to a width of 1.5 picas:

```
<savetext textid="listmrk" conrule="seqlstct,\.\,@1.5pi">
```

(Then, if the item has a First Line indent exactly 1.5 picas less than its Left Indent, the initial letter of all lines will align vertically provided the item number does not exceed a width of 1.5 picas.)

To save the content of the section title (which is the current e-i-c) for eventual processing as part of the table of contents:

```
<savetext
    textid="toc"
    append="1"
    conrule="<toc2>,#CONTENT,<tocpg>,folioct,</tocpg>,</toc2>">
```

This assumes an e-i-c entry in the FOSI for the “toc2” pseudo-element whose associated formatting characteristics are set to be appropriate for a section title entry in the table of contents, an e-i-c entry in the FOSI for the “tocpg” pseudo-element whose characteristics help set a page number in the table of contents, and a FOSI-defined counter called “folioct” whose value is the current page number. This entry, as well as all others saved with textid="toc" provided they all have append="1", would be available for retrieval by a Usetext.

Considering the above example, if it were desired at some point to forget all that has been saved to textid="toc", the following Savetext construct could be used:

```
<savetext
    textid="toc"
    append="0"
    conrule="">
```

Assume that the following usetext appears in the header of a page of a dictionary, where the e-i-c for each word in the dictionary contains a savetext to the textid `currword`:

```
<usetext source="currword[FB]">
```

Also assume that the e-i-c for the `word` element in the dictionary contains a savetext similar to:

```
<savetext textid="currword" conrule="<hword>,#CONTENT,</hword>">
```

This combination of savetexts and a usetext will produce a list in the header of all of the words where some part of their definition appears on the page. The e-i-c for `hword` could specify any special formatting the document design requires.

B.3.5 FORMATTING OUTPUT SPECIFICATION INSTANCE (FOSI) VARIABLES. To assist with some of the more complex style issues, such as enumeration and reuse of text, the OS defines two kinds of constructs that act as variables within the FOSI. One kind is the counter type which contains numeric values that can be incremented. The other kind is the string type (also known as textids since these variables are referenced by the savetext category's textid characteristic) which contains strings and other constructs as described by the section on construction rule syntax B.3.4.10. Variable names are referenced in various OS attributes, some of which may have an SGML declared value of CDATA which allows mixed case; however, all FOSI variables shall be recognized and processed in a case-insensitive manner.

Note that FOSI variables such as counters and previously saved text are expanded (that is, their values are determined) at the time they first appear in the Construction Rule. However, the context of any elements (that may have been saved within #CONTENT) or pseudo-elements that get saved or used in a construction rule, which will be used to determine which e-i-c to match in the FOSI as well as parentage for inheritance, shall be based on the context in which the saved text is finally used (via a Usetext).

A variable can be time dependent or time independent. Time independent implies that the variable has a constant value throughout the processing of the variable's scope (though, in effect, this is often only implementable by making some kind of initial pass through the document to calculate its effective value before final processing is possible). Counters are always time dependent; that is, its value at any particular point in the document is the value last saved into it. If no save has yet been done to the counter, its value is the value set for the counter in the resource description. String variables can be declared to be either time dependent or time independent (using the stringdecl element in the resource description). If a string variable has no declaration, it is time dependent except in the special case of a textid being filled in from a source attribute value via a fillval in which case the variable is time independent

Time independence allows for forward references to work (although how this is accomplished is left to the FOSI/output system implementation). If a FOSI Savetext variable is time independent, its value is assumed to be available throughout its scope. If it is time dependent, its value at any particular point in the document is the value last saved into it. If no save has yet been done to the variable, its value is the value set for the variable in the resource description if any, or else the null string. A FOSI Savetext variable is time independent if the value of its Time Status attribute in the resource description is enabled; otherwise, the variable is time dependent except in the case that the variable name has been filled in from a source document attribute via a fillval in which case it shall be time independent. The time-independent value of the variable is its value at the end of the first element instance encountered which is specified by the variable's Scope attribute in the resource description (or else at the end of the document element). A

time-independent variable will be reset when the end tag of any element in the variable's scope is encountered and after all references to the variable within the scope have been resolved. If any element in the variable's scope is a child of another element in the scope, the result is left to be determined by the output system.

String variables may be defined to be "exportable" when they are declared (see B.4.1.4). Variable names are generally scoped by their module (ospackage see B.3.15).

B.3.6 SPECIFYING ELEMENTS-IN-CONTEXT (E-I-CS).

B.3.6.1 General. In a FOSI, characteristics must be specified for each element in every context in which a formatting system needs to treat the element differently. That is, before characteristics can be attached to elements, the class of elements to which a given set of characteristics will apply must be completely specified. This is accomplished by providing the following information:

Generic Identifier (GI)	This is a unique name that identifies an element. It may exist in the source DTD or be defined as a pseudo-element. For a more detailed description see ISO 8879.
Context	The context attribute gives the lineage of the GI. This specifies a context(s) in which the element may appear.
Occurrence	The order of appearance of this e-i-c in relation to like elements. All occurrences of an e-i-c within its parent, regardless of intervening elements of a different type, are considered part of one e-i-c group. For example, when a list contains item(1), item(2), note(1), and item(3), item(1) is the "first" item and item(3) is the "last" item.
GI type	When the value of this GI type attribute is "element", as is its default, the value of the GI attribute is taken to be the name of an element (in the source DTD) or pseudo-element (elements and pseudo-elements share the same name space). However, by setting GI type to "pi", the value of the GI attribute is taken to be the name of a processing instruction to which this e-i-c will apply.

B.3.6.2 Context. The characteristics an element takes on may be a function of the particular lineage of ancestors it has when it occurs. For example, "title" has many uses and must be specified in context with its parent such as "Chapter" or "para0". In this case additional e-i-c entries may be necessary if the formatting requirements for the context of chapter are different than that of para0. To specify the context of an e-i-c, the syntax described in B.3.6.2.1 and B.3.6.2.2 shall be used.

B.3.6.2.1 Context attribute syntax. The context attribute gives the lineage of the GI. The context characteristic is a string of one or more GI(s) or wildcards separated by spaces.

A wildcard is specified by a single asterisk. A special case is the GI #PAGEDESC (case-insensitive)—which is only valid as the last (or only) token in a context string—which will cause an element instance to match this context only if the element instance is being resolved for formatting in the page description part of the FOSI. The first (leftmost) GI specifies the parent or most closely related ancestor of the element. Moving from left to right, subsequent GIs must be parents or ancestors of the previous GI. (For the purposes of context, an element containing an entity reference to a SUBDOC entity in its content or as the value of an attribute with declared value of ENTITY shall be considered the "parent" of the document element in the SUBDOC entity.) The wildcard can be used to specify zero or more levels of ancestry that are not referenced by name. When more than one wildcard appears in sequence in the context, it is considered to be one wildcard. The value of the context attribute consists of any number of complete context strings separated by a semi-colon (";") and optional spaces surrounding each semi-colon. When more than one context string is provided, each context is used independently of the others; that is, an e-i-c with multiple contexts is merely a specification short cut equivalent to giving multiple e-i-cs with the same charlist and attspec set but each with a different one of the context strings. Examples of context attribute specifications:

A chapter title: `gi = "title" context = "chapter"`

A figure title in the body: `gi = "title" context = "figure * body"`

A list within a warning: `gi = "seqlist" context = "warning"`

A list within a warning within a primary paragraph: `gi = "seqlist" context = "warning * para0"`

A list within a warning within front matter: `gi = "seqlist" context = "warning * front"`

A list item within a list within a list within a list in a section: `gi = "item" context = "seqlist * seqlist * seqlist * section"`

Any footnote in the front matter: `gi = "ftnote" context = "front"`

A footnote in a table: `gi = "ftnote" context = "* table"`

A title of any one of an intro, a tasklist within a procedure, or a testproc: `gi = "title" context = "intro; tasklist procedure; testproc"`

B.3.6.2.2 Best matching e-i-c. Best match describes how to choose the e-i-c to use for any particular occurrence of an element within the document.

B.3.6.2.2.1 Terms. The following terms are used:

a. CURRENT ELEMENT is the element of interest in the document.

b. GI is the value of the generic identifier characteristic specified for the e-i-c in a FOSI.

c. CURRENT PATH represents the current hierarchy in the document. It can be thought of as an ordered list of element names beginning with the parent of the

current element, followed by its parent, and so on, terminating with the document element.

d. CONTEXT PATH represents a possible hierarchy that could appear in the document (based on the content models of the source DTD). It can be thought of as an ordered list of element names and wildcards. It is specified as the "context" characteristic for an e-i-c in a FOSI.

e. NODE is a particular element name or wildcard in a path.

f. CURRENT NODE is the node being currently examined in a path.

g. CANDIDATE LIST is the list of e-i-cs that are possible choices (candidates) for being the best match.

B.3.6.2.2.2 Best match algorithm. The goal of the following algorithm is to determine which CONTEXT PATH most specifically matches the CURRENT PATH. The algorithm has two parts: first determining which e-i-cs are possible matches (creating the CANDIDATE LIST), then determining which one is the best path. A candidate list is first created without assuming any implicit initial wildcards, and then, only if that candidate list produces no matches, a new candidate list is made assuming implicit initial wildcards. The main algorithm for creating the CANDIDATE LIST is as follows:

a. Put on the CANDIDATE LIST all the e-i-cs whose GI matches the CURRENT ELEMENT (and whose occurrence specification applies to the CURRENT ELEMENT). Repeat steps b and c for each candidate.

b. Make the leftmost node the CURRENT NODE of both the CURRENT PATH and CONTEXT PATH.

c. Does the subpath of the CONTEXT PATH starting with its CURRENT NODE match the subpath of the CURRENT PATH starting with its CURRENT NODE? (See following subalgorithm.) If not, remove the e-i-c from the CANDIDATE LIST.

d. If no candidates remain, put on the CANDIDATE LIST all the e-i-cs whose GI matches the CURRENT ELEMENT (and whose occurrence specification applies to the CURRENT ELEMENT). Repeat steps b and c for each candidate, assuming that the CONTEXT PATH for each candidate contains an initial wildcard. For example, "chapter body" would now be interpreted as "* chapter body".

The subalgorithm to determine if the subpath of the CONTEXT PATH starting with its CURRENT NODE matches the subpath of the CURRENT PATH starting with its CURRENT NODE is as follows:

a. If the CURRENT NODE of the CONTEXT PATH is a wildcard then:

   1. Move right in the CONTEXT PATH until the CURRENT NODE is not a wildcard or the end of the path is reached. If the end of the path is reached, the paths do match; exit this subalgorithm.

   2. If the CURRENT NODE of the CONTEXT PATH does not match the CURRENT NODE of the CURRENT PATH, move right in the CURRENT

PATH until its CURRENT NODE is the same as the CONTEXT PATH's CURRENT NODE or the end of the CURRENT PATH is reached. If the end of the path is reached, then the paths do not match; exit this subalgorithm.

3. Does the subpath of the CONTEXT PATH starting with the node to the right of the CURRENT NODE match the subpath of the CURRENT PATH starting with the node to the right of the CURRENT NODE? If not, make the next node to the right in the CURRENT PATH the current node and go back to the previous step. If the subpaths match, then the paths are a match; exit this subalgorithm.

b. If the CURRENT NODE of the CONTEXT PATH is not a wildcard:

1. If the CURRENT NODE of the CONTEXT PATH matches the CURRENT NODE of the CURRENT PATH:
   (a) Make the next node to the right the current node in both paths.
   (b) If we have reached the end of the CONTEXT PATH, then the paths do match; exit this subalgorithm.
   (c) If we have reached the end of the CURRENT PATH then the paths do not match; exit this subalgorithm.
   (d) Go to step a of this subalgorithm.
2. At this point, the paths do not match; exit this subalgorithm.

Notice that this subalgorithm calls itself recursively in step a.3, and the phrase "exit this subalgorithm" always means to exit "up one level" to the calling algorithm.

B.3.6.2.2.3 Default match. If the candidate list produces no matches (that is, if there is no e-i-c entry in the FOSI for this source element), the result is as if there were a matching e-i-c entry with an empty charlist.

B.3.6.2.2.4 Best match evaluation. A unique best match is selected from the final candidate list by using the following algorithm. For each member in the candidate list, "line up" the nodes of the context path with the matching nodes in the current path (where a wildcard in the context path may "line up" with any number [possibly zero] of consecutive nodes in the current path). For each member, form a string by reading the expanded context path left to right and: (1) for every explicit node in the context path, append a 1 and (2) for every wildcard in the context path, append one more 0 than the number of nodes it matches. For example, if an asterisk in the context path matches 3 nodes in the current path, 4 zeros should be appended to the string. An initial decimal point should be inserted at the beginning of each string. These strings are used as the match score for the given member. The candidate list with the highest score shall be the best match. Note that this algorithm puts more emphasis on matches near the beginning of the list and guarantees a unique result among a set of distinct candidates.

For example, consider the following current path and candidate list:

CURRENT PATH:

```
"para  item  randlist  para  step1  para0  chapter"
```

CANDIDATE LIST:

```
"para"
"para item"
"para * item"
"para * para0"
"para * randlist * chapter"
"* para step1 * chapter"
```

The results of “lining up” candidates with the current path are as follows (the number following each wildcard indicates the number of nodes in the current path it matches):

CURRENT PATH:

```
"para      item  randlist     para  step1     para0  chapter"
```

CANDIDATE LIST:

```
"para                                          " = .1
"para      item                                        " = .11
"para * 0  item                                        " = .101
"para * 4                                     para0    " = .1000001
"para * 1        randlist * 3                        chapter" = .100100001
"* 3                          para  step1 * 1      chapter" = .000011001
```

Evaluating each string as a number between zero and one we obtain a best match score of .11 for context="para item".

B.3.6.2.2.5 Implementation notes. Note that this algorithm produces absolute rather than relative values so that the determination of the match score of any candidate can be made before the complete candidate list is determined. In fact, the best match can be computed in parallel to the creation of the candidate list, and some potential candidates can be eliminated before being put on the candidate list by virtue of their match score being less than that of the best candidate to date.

B.3.6.2.2.6 Priority. This best match algorithm determines a unique match among all the possible distinct context strings. However, if several e-i-cs have identical context strings and differ only by their occurrence attribute, the absolute best match will be that e-i-c with the highest priority occurrence value. If there is more than one e-i-c with the exact same context and occurrence in a given FOSI module, the first e-i-c encountered is the one that is used. For precedence between multiple modules (“osmodules”), see B.3.15.

B.3.6.3 Occurrence. The occurrence attribute of the e-i-c element further refines the context in which this e-i-c is used by specifying the order of appearance of this e-i-c in relation to like elements with the same parent. All element instances with the same GI and the same parent element form a group out of which the occurrence indication can choose a subset. The possible occurrence values, shown in descending order, have the following meanings:

a. Only: the element instance in question will match this e-i-c only if it is the only element (with this GI) in the group.
b. First: only the first element in the group will match this e-i-c. (If the group only has one element, it will qualify as the First.)

c. Last: only the last element in the group will match this e-i-c. (If the group only has one element, it will qualify as the Last.)

d. Middle: all elements except the first and last in the group will match this e-i-c. (If the group has less than three elements, no element in the group will qualify as the Middle.)

e. Notlast: all elements except the last in the group will match this e-i-c. (If the group only has one element, it will not qualify as the Notlast. If the group only has two elements, the first one will qualify as the Notlast.)

f. Notfirst: all elements except the first in the group will match this e-i-c. (If the group only has one element, it will not qualify as the Notfirst. If the group only has two elements, the last one will qualify as the Notfirst.)

g. All (the default): all elements in the group will match this e-i-c. Note that a given element instance may qualify for more than one occurrence value in which case the best match is determined by the priority of the occurrence values for which it qualifies.

B.3.6.4 GI type. The GI type attribute (gitype) indicates whether the value of the e-i-c's GI attribute gives the name of an element (or pseudo-element) or the name of a special class of processing instruction (PI) recognized by FOSI-processing systems (see B.3.6.7). If the GI type attribute is not assigned, the GI is assumed to be an element (or pseudo-element); for the GI value to refer to a processing instruction, the GI type attribute must be set to "pi" (for example, gitype="pi"). It is possible to give the same name to both an element and a PI without there being any confusion since the GI type attribute provides the mechanism for maintaining the distinction between these two name spaces.

B.3.6.5 Source document attributes.

B.3.6.5.1 General. The charlist should be written to specify the formatting intent for an element that will have no attributes attached to it (for example, <para0> versus <para0 tocentry="1" label="INTRO">). The value assigned to attributes through the SGML markup may affect the characteristics assigned to an e-i-c. There are three possibilities:

a. The attribute has no effect on formatting. For example, an attribute in the source document which provides identification, such as part number.

b. The attribute's value affects which characteristics are activated, but does not specify the value of the characteristic. For example, the value associated with an attribute used to indicate if a table of contents entry is desired for an element in the source document. Such an attribute is used by the FOSI to either extract or not extract the corresponding content to the table of contents.

c. The attribute's value must be used as a characteristic value. In other words, the attribute value is directly assigned to a characteristic value. For example, from the source document, an attribute used to indicate the number of columns in a table supplies that value to the FOSI for use in determining the final appearance of the table.

An attribute specification (also called “attspec” and indicated in a FOSI by the `att`, `attif`, `attelseif`, and `attelse` elements) allows the user to qualify the formatting intent based upon possible attribute occurrences. In other words, based upon the appearance of a certain attribute value or by extracting an attribute value from the source instance the user is permitted to specify additional formatting intent or override the characteristics that were specified in the original charlist. The mechanisms for handling these cases (with the exception of the first which requires no special treatment) are specval and fillval. These mechanisms, used alone, or together, enable the FOSI author to specify the attribute(s) that affect characteristic values specified in the resulting charlist.

An attribute specification logically consists of two basic parts: the conditional part, consisting of zero or more occurrences of specvals and fillvals, and the charsubset whose contents will affect the formatting of the e-i-c if and only if the conditional part is satisfied. If the conditional part is omitted, the attspec is known as an unconditional attspec, and its charsubset always takes affect. (Though characteristics that always take effect are generally put into the e-i-c’s main charlist rather than in an unconditional attspec, the latter construct is useful when some unconditional formatting must occur in a specific order relative to other possibly conditional attspecs.) The conditional part can combine multiple specvals and fillvals. Depending on the setting of the att element’s logic attribute, the entire conditional part can be satisfied either only when all individual specvals and fillvals are satisfied or whenever any one of the specvals or fillvals are satisfied (see B.3.6.5.4).

Structurally, an attspec can be either a simple `att` element or an if-then-else combination contained within an `attif` element. A simple attspec contains a single condition (consisting of zero or more occurrences of specvals and fillvals) and a single charsubset. The charsubset takes affect if the condition is satisfied. An if-then-else attspec consists of multiple condition/charsubset pairs organized by subelements (`attelseif` and `attelse`) of the `attif` element, wherein if the condition of the initial condition/charsubset pair immediately contained in the `attif` element is satisfied, then its associated charsubset takes effect and the entire remainder of the if-then-else attspec is ignored. If this condition is not satisfied, then the condition of each condition/charsubset pair associated with each subsequent `attelseif` element in order (if any) is tested and if it is satisfied, its associated charsubset takes effect and the entire remainder of the if-then-else attspec is ignored. If none of these conditions are satisfied, then the charsubset associated with the `attelse` element, if present, takes affect. In the following discussion of specvals and fillvals, the text and examples generally refer to specvals and fillvals used in the context of a simple attspec, but they apply equally well to specvals and fillvals used within if-then-else attspecs.

B.3.6.5.2 SPECVAL.

B.3.6.5.2.1 General. This feature is used for the case where the attribute affects the use of a characteristic but does not supply the value of the characteristic. The charsubset associated with the SPECVAL will take effect when the attribute specified by attname contains the value specified by attval. It has the following characteristics associated with it:

a. ATTNAME. The name of the source attribute that is being identified as having an effect on formatting. When the attname value does not specify the name of a valid attribute, the specval will be considered to be unsatisfied and the associated characteristic values for this attribute specification are not applicable.

b. ATTLOC. The name of the element on which the attribute was specified. If unspecified, it is assumed to be the same as the "GI" for the e-i-c being described. If specified, the name should be the name of an ancestor of the e-i-c and indicates the most immediate previous ancestor of that name. This allows referencing the values of ancestors' attributes to determine formatting of the current element. Alternatively, this field will contain the keyword #FOSI to indicate that ATTNAME is the name of an internal FOSI variable rather than a source attribute. When the attloc value does not specify the name of a valid ancestor of the current e-i-c or the attname value does not specify the name of a valid source attribute for this ancestor, the specval will be considered to be unsatisfied and the associated characteristic values for this attribute specification are not applicable.

c. ATTVAL. The value for the attribute identified in "ATTNAME" which results in a certain formatting difference.

A simple use of specval in an attribute specification would have the following basic structure:

```
<e-i-c gi="element">
  <charlist>
    characteristics without respect to attributes

  <att>
    <specval
      attname="name of source attribute (on element specified by attloc)"
      attloc="name of an ancestor of an element"
      attval="value contained in attname">
    <charsubset>

      characteristics to take effect if specval is satisfied.
```

B.3.6.5.2.2 Evaluating a specval. A specval is considered to be satisfied when the source attribute specified by attname (and optionally attloc) has been assigned the value indicated by attval. If an attspec is satisfied, the associated characteristic values (from the charsubset) must be "merged" into the charlist for this e-i-c. Most categories (for example, font) can only occur once in a charlist, but some (for example, savetext, puttext) may occur multiple times and "merging" has a different meaning in these two cases.

B.3.6.5.2.3 Specval syntax. For a specval, the value for attval should represent one of the possible values that an attribute can have. In addition, the following key words can be used:

a. #NONZERO may be supplied for attributes whose declared value is a number (not CDATA) to indicate any value that does not consist of zero or more blanks followed by one or more zero characters followed by zero or more blanks.

b. #NONE indicates that no assignment was made to that attribute in the source document. Note that this condition is possible only when the attribute has a default value of #IMPLIED.

c. #ANY may be used for any attribute to indicate that an assignment was made in the source document.

d. #ITEM#YYY may be used to indicate that the value of attval contains one item from a list, where YYY is a literal. For example, #ITEM# would be used for an attribute with a declared value of NAMES. In this case the characteristics specified for all of the items for one attribute are cumulative where possible; all the values take effect instead of overriding one another. One case may be highlighting, which has numerous formatting options available.

e. #XX#YYY may be used for attributes whose declared value is a number where XX is one of the letter pairs LT, GT, EQ, LE, GE, NE and YYY is a numeric constant or counter id (enumid). In this case, the specval will be satisfied if the value assigned to the given source document attribute is, respectively, less than, greater than, equal to, less than or equal to, greater than or equal to, or not equal to the given numeric constant or current value of the given counter. The same syntax may be used for attributes whose declared value is not a number, but in this case XX can only be EQ or NE, and YYY is a backslash delimited literal or a savetext name. In this case, if the declared value of the attribute is CDATA, the comparison will be case sensitive, otherwise it will not be case sensitive.

When the referenced source attribute has no assigned value (that is, when it is left #IMPLIED), all these tests will fail except for the explicit test against #NONE which will succeed. In particular, #NONZERO and #NE#yyy and #EQ#\\ will all fail when the referenced source attribute is unassigned. If the referenced source attribute is not valid (on the current element if attloc is unspecified or on the element specified by attloc) according to the source DTD, then this specval will fail in all cases (including the #NONE test).

B.3.6.5.2.4 Merging charsubsets into charlists. A charsubset is a named collection of characteristic value assignments that can be merged into the charlist of an e-i-c. Charsubsets are merged either by reference via a charlist's charsubsetref attribute (or indirectly via a referenced charsubset's charsubsetref attribute) or by satisfaction of an attribute specification.

The ordered steps to produce a merged characteristic list from referenced charsubsets, the charlist, and attspecs are:

a. All charsubsets referenced by the charlist's charsubsetref of an e-i-c are merged in the order referenced.

b. All categories specified in the charlist are merged into the result of the previous step; then inheritance and defaulting occur (see B.3.7).

c. Charsubsets that are part of successful attspec are merged in order into the result of the previous step.

The charlist and charsubsets are merged according to the following rules:

a. If the charlist/charsubset contains a category that is permitted zero or one time in the charlist by the OS DTD, that category's characteristic's settings in the current charlist/charsubset "merge" into the merged list by setting the value of the characteristic in the merged list to that explicitly specified in the current charlist/ charsubset. (Note: only those characteristics explicitly specified in the current charlist/charsubset's category will merge into the merged list; characteristics in the same category that are not explicitly specified in the charsubset's category are not affected.)

b. If the charlist/charsubset contains a category that is permitted zero or more times in a charlist by the OS DTD, that category (with all its explicit settings) is appended to all other occurrences of repeatable categories in the order in which they are encountered.

Note that the order of processing is significant: for non-repeatable categories, a characteristic will get its value from the "last" item providing a value, and for repeatable categories, the order of the collection of all repeatable categories will reflect the order of specification and merging.

B.3.6.5.3 FILLVAL.

B.3.6.5.3.1 General. This feature is used for the case where the attribute's value is used as a characteristic value. For this case, additional characteristics for the corresponding category may be filled in the charsubset immediately following the fillval (or sequence of fillval) category(ies). It has the following characteristics associated with it:

a. ATTNAME. Same as SPECVAL.

b. ATTLOC. Same as SPECVAL.

c. FILLCAT. The OS category for which the source attribute has an effect.

d. FILLCHAR. The characteristic found in the category specified in "FILLCAT" for which the value from the source attribute is to be assigned.

e. CONRULE. Rule for specifying the construction of the filled value.

Fillval has the following basic structure:

```
<e-i-c gi="element">
  <charlist>
    characteristics without respect to attributes

  <att>
    <fillval
      attname="name of source attribute (on element specified by attloc)"
      attloc="name of a direct ancestor of element"
      fillcat="charlist category name to be used"
      fillchar="category characteristic name to fill value into">
    <charsubset>
      all remaining characteristics that are affected by the appearance of the source attribute.
```

B.3.6.5.3.2 Evaluating a fillval. A fillval is considered to be satisfied when the source attribute specified by attname (and optionally attloc) has been assigned a value in the source instance (either explicitly or due to an explicit default being specified in the source DTD). As with a specval attspec, if a fillval attspec is satisfied, the associated characteristic values (from the charsubset) must be "merged" into the charlist for this e-i-c. Most categories (for example, font) can only occur once in a charlist, but some (for example, savetext, puttext) may occur multiple times, and "merging" has a different meaning in these two cases. In the case of a fillval, the value of the attribute may also get merged into the charsubset (as specified by the fillcat and fillchar) before the charsubset is then merged into the e-i-c's charlist (see B.3.6.5.2.4).

B.3.6.5.3.3 Fillval syntax. For a fillval, the value of fillcat should be the name of a characteristic category (one of the GIs in the content model for "charlist" in the OS DTD or appropriate table, figure, and graphics characteristics) and the value for fillchar should be the name of a characteristic (one of the attributes associated with those GIs in the OS DTD). If no Construction Rule is specified, the source attribute value is assigned to the specified characteristic with no modification. When a Construction Rule is specified, the source attribute value is designated by #CONTENT, and the resolved Construction Rule is assigned to the specified characteristic (see B.3.4.10).

B.3.6.5.3.4 Merging the attribute value into the fillval's attspec. For each satisfied fillval in a given attspec, the specified attribute's value is filled into the first explicit occurrence in the charsubset of the fillcat category in which the fillchar characteristic is not explicitly assigned. (Note that any charsubsets that may be referenced via the charsubsetref attribute on the charsubset element are ignored for the purposes of determining how to merge the fillval into the attspec's charsubset.) If there is no explicit occurrence of the fillcat category in the charsubset, the fillval will get merged into the charsubset as if there had been an explicit occurrence of the fillcat category with no characteristics assigned as the last element in the charsubset.

B.3.6.5.4 Multiple fillval and specval specifications. When there are multiple specvals and fillvals contained in a single attspec, all of the specvals and fillvals in the attspec are first evaluated, then the value of the logic attribute is evaluated to determine what conditions must be met in order to satisfy the attspec. When the logic attribute is "or", at least one of the specvals or fillvals must be satisfied in order for that attspec to be satisfied. When the logic attribute is "and", all the specvals and fillvals in the attspec must be satisfied in order for that attspec to be satisfied. If the attspec is satisfied, first the satisfied fillvals (if any) get merged into the attspec's charsubset, then the charsubset is merged along with other satisfied attspecs into the charlist.

B.3.6.6 Pseudo-elements. Generally the GI of an e-i-c entry in the FOSI identifies an element in the source DTD. However, the GI may also name a "pseudo-element" that does not conflict with the name of any element in the source DTD. Pseudo-elements can be included in the Construction Rule of a Savetext or the Source of a Usetext. Whenever a pseudo-element appears in a Usetext Source or a Savetext Construction Rule, both the start and end tags of the pseudo-element must be present. The formatting characteristics

would take effect when the pseudo-element is used via the Usetext category. Note that a pseudo-element does not appear in the source DTD, but only in the FOSI. The pseudo-element is a construct that can act as a reference to an e-i-c entry in the FOSI. There are a number of important exceptions to the ability of pseudo-elements to act like elements from the document instance. Pseudo-elements have no content model. They do not have attributes and therefore cannot appear in a specval or fillval attloc. All charlist categories work with pseudo-elements in the same manner as they do with elements from the document instance.

B.3.6.7 Specifying an e-i-c for processing instructions. The e-i-c element has a “gitype” attribute whose value defaults to “element” to indicate that the GI value refers to an element (or pseudo-element) name. However, if the “gitype” attribute’s value is set to “pi”, then the value of the GI attribute is interpreted to refer to the “name” of a processing instruction (PI) as defined in the following paragraphs. This feature allows an e-i-c entry to attach formatting characteristics to a special class of processing instructions flagged by the keyword “FOSI” following the “<?” of the PI.

A PI e-i-c actually associates formatting with the part of the input document delimited by two different but related PI’s that are collectively known as a “paired PI”, whose composite individual PI’s function analogously to the start and end tag for an element. For example, to use PI’s to markup a region in which hyphenation is inhibited, the instance might be tagged with <?FOSI hyphoff> and <?FOSI /hyphoff>. The FOSI would then contain an e-i-c whose gi="hyphoff" and gitype="pi" and whose charlist contained <hyphen hyph="off">.

With respect to attspec processing in PI e-i-cs, all “attribute” matching would behave as with element e-i-cs, and the assumption would be that all PI “attributes” are treated as CDATA type. The text of the PI following the PI’s “name” should parse as an SGML start tag’s attribute value list would parse, and constructs that look like “attribute value assignments” would be able to be used in the PI e-i-c’s attspec. Note, however, that a PI e-i-c’s attspec’s attloc value must refer to the name of a DTD element (or #FOSI); a PI “name” would not be recognized as an attloc value.

By definition and by analogy with DTD element e-i-cs, the affect of any characteristic changes would be scoped by the PI e-i-c’s “start” and “end” PI’s. These “paired” PIs are required to be well-nested with respect to themselves, other paired PI’s, and the element structure of the instance. If a start PI is encountered such that a matching end PI is not found within the appropriate scope, then the start PI will be treated as if an omitted end PI occurred immediately preceding the close of the scope that encloses the start PI (and any unmatched end PI would simply be ignored). Note that a start PI immediately followed by an end PI can be a reasonable thing to do: some examples are a PI e-i-c that forces a page break or one that does a puttext, or savetext. But to have an effect on the ongoing environment (such as a change to the font, indent, or quadding), a paired set of PI’s would be required since otherwise the (potentially implicit) end PI that follows the start PI terminates the scope of any changes made by the start PI.

Specifications for the “context” and “occur” attributes on the e-i-c element for PI e-i-cs function just as for DTD element e-i-cs. Note however that only element and pseudo-element names can be specified within a context string (that is, PI “names” are not permitted). That is, PI e-i-cs can have context, but do not create or contribute to the current context for either themselves or for element e-i-cs as far as e-i-c matching is concerned. (The word “context” is used here in the strict sense of e-i-c matching, and not in the sense of composition environment, which is discussed in the next paragraph.)

The affect PI e-i-cs have on the composition environment is identical to that of element e-i-cs. In particular, inheritance of OS characteristics operates equivalently with respect to both. For example, if a PI pair has changed the font to 10pt and then an emphasis element that is wholly contained within the PI pair’s scope inherits the font category but sets the posture to italic, the character data within the emphasis element will inherit the size of 10pt as set by PI pair.

B.3.7 INHERITANCE AND DEFAULTING. There are many characteristics that can potentially be specified in a charlist. In practice, however, most e-i-cs omit explicit specifications for many of the characteristics. The use of inheritance within a FOSI provides a means for avoiding the explicit specification of every characteristic value for each in cases where the value seldom changes from one element to the next. Inheritance and defaulting provide mechanisms for determining values for unspecified characteristics in an e-i-c’s charlist.

Note that inheritance and defaulting occurs relative to an e-i-c’s “resolved charlist.” This is the logical charlist that results after merging of all appropriate attribute specifications (specval’s and fillvals) and charsubsets. Sub-characteristic lists (subchars) usually specify only a small set of characteristics. Although in a strict sense, inheritance and defaulting does not apply to subchars, the mechanism by which the remaining characteristics that are associated with the subchars are determined is quite similar to the inheritance and defaulting rules for charlists. Also note that the inherit attributes of the various categories are not characteristics and are never inherited or defaulted.

When a characteristic has an explicit assignment in a charlist, this determines the value of the characteristic (though in some special cases—for example, keeps, suppress, prespace, and postspace—the determined characteristic value may be ignored or modified due to the particular definition of the effect of the characteristic).

When a characteristic has no explicit assignment in a charlist, how its value is determined differs depending on whether its category is inheritable (that is, includes an inherit attribute) or not. An inherit attribute can have the value “off” (zero) to indicate that inheritance is not enabled for this category occurrence or a value of “on” (non-zero) to indicate that inheritance is enabled for this category occurrence. If a category’s inherit attribute is not explicitly assigned, its value will default to the value of the charlist’s inherit attribute (which always has a value since the OS DTD assigns it a default value of “off”).

a. For inheritable categories:

1. If an inheritable category is omitted entirely from a charlist, the effect should be as if the category were specified with no explicit characteristic assignments, but with the inherit attribute set to the value of the (possibly defaulted) charlist's inherit attribute. That is, when an inheritable category is omitted from the charlist, if the charlist inherit is "on", inheritance is enabled for this category (see paragraph a.2.), whereas if the charlist inherit is "off", inheritance is not enabled for this category (see paragraph a.3.).
2. If inheritance is enabled for this category (that is, the value of the inherit attribute is set "on" either explicitly or by virtue of defaulting to the value of the charlist inherit attribute), the characteristics in this category that are not explicitly assigned values inherit their respective values from the parent element instance in the source document.
3. If inheritance is not enabled for this category, the characteristics in this category that are not explicitly assigned values obtain their respective values from the environment indicated by the environment name (envname) attribute of the charlist. If no environment is specified, the values are obtained from the global default environment (first docdesc encountered when "modules" are merged), unless a local docdesc has been defined.

b. For uninheritable categories:
   1. If an uninheritable category is omitted entirely from a charlist, no processing relative to that category occurs, and the values of its characteristics are unaffected.
   2. If the category appears in the charlist, characteristics not explicitly assigned values obtain their respective values from the environment indicated by the environment name (envname) attribute of the charlist. If no environment name is specified, the values are obtained from the default environment (docdesc).

c. If after inheriting and defaulting as specified, there is still no value assigned to a characteristic, then either that characteristic has no affect in the current situation or its value is left to be resolved by the output system. (Certain values, such as that for letterspace, wordspace, or extra end-of-sentence space, may be best optimized by the output system.)

In sub-characteristic lists (subchars), all unspecified categories that can be inherited behave as though the category was specified with its Inherit attribute enabled (that is, a Subcharacteristic List behaves in the same way as a Characteristic List whose Inherit attribute is enabled). Each specified inheritable category inherits or defaults (depending on the setting of its Inherit attribute as described in step a); sub-characteristics that are inherited take the resolved value for this characteristic from the encompassing charlist. All categories that cannot be inherited default to the same environment to which the containing charlist defaults (as described in step b). For the Change Mark and ruling categories which have non-empty content other than subchars, inheritance and defaulting work the same as with subchars; that is, the rules for inheritance and defaulting are as if the categories in the content of Change Mark and Ruling were wrapped by a subchars element.

Inheritance on the outermost (doc) element is interpreted as being equivalent to defaulting to the default environment in effect for that tag. Inheritance on the document element of a subdoc entity is permitted; for the purposes of inheritance, the “parent” of the subdoc’s document element is the element “containing” the entity reference to the subdoc entity (either as part of its content or as the value of an attribute).

A charlist subset (charsubset) has both charsubsetid and charsubsetref attributes. These attributes provide for referencing specific charsubsets in charlists, atts, or other charsubsets, for example. The use of charsubsets in a charlist allows for replacing specific characteristics, while still inheriting characteristics from the parent.

An environment description (envdesc) contains a charlist which has “inherit” and “envname” attributes. The “inherit” attribute on the charlist in an envdesc will be ignored; that is, it will always be treated as if it were set to zero “do not inherit”. All the inherit attributes on the categories in an envdesc’s charlist are also ignored. The “envname” attribute on the charlist in an envdesc can specify the name of an already declared envdesc; any characteristic not explicitly specified in an envdesc will be given the value from the default environment named by “envname” (which must have been declared previously in the FOSI). Note that if the “envname” attribute is not specified on the charlist of the envdesc, the envname would default to the docdesc. Therefore any characteristic in a category that is not specified for this environment would take on the values as assigned in the default environment. Note, since an environment description provides defaults for unspecified characteristics, it is inappropriate for any category to appear more than once in the same environment description.

APPENDIX B

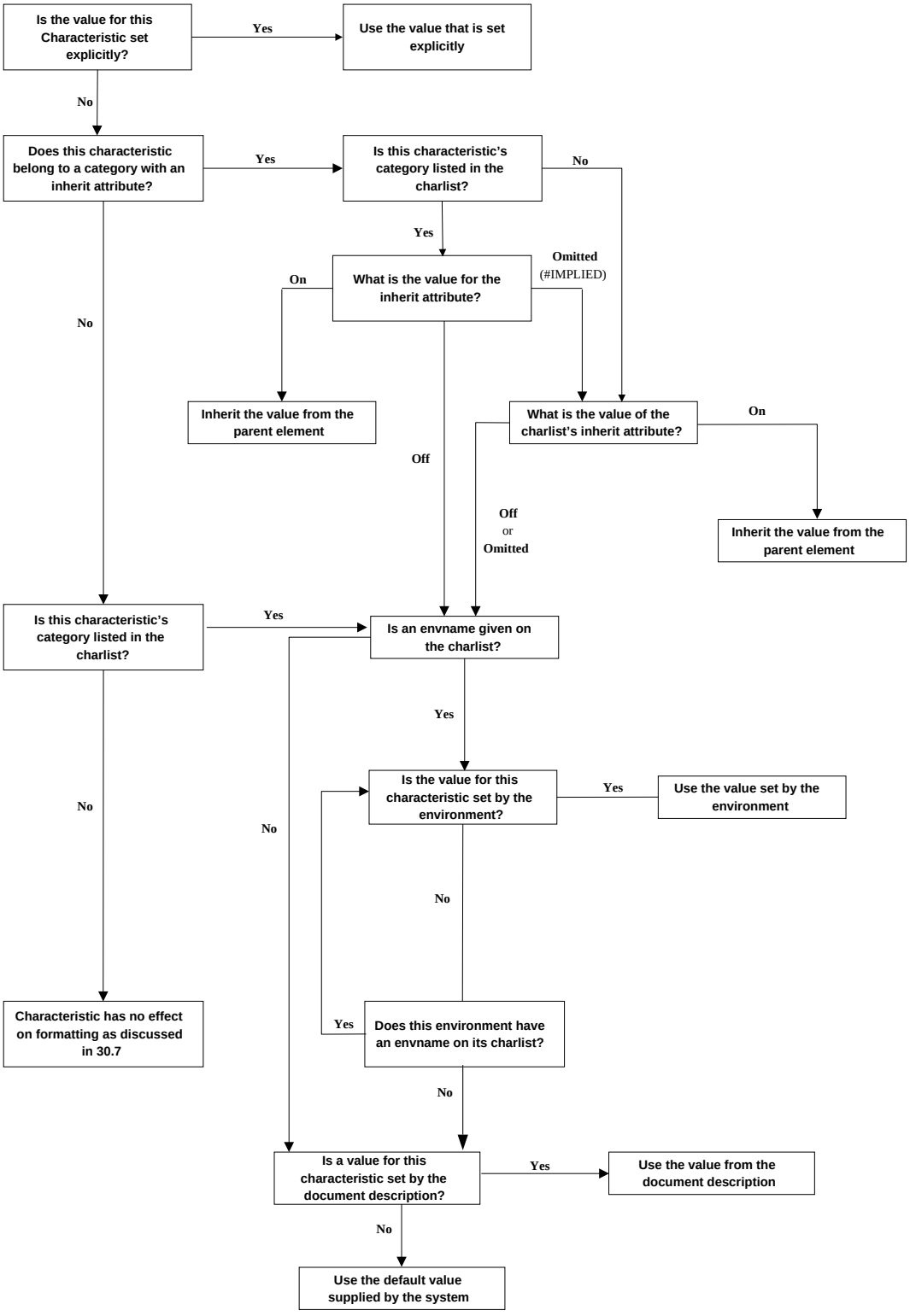

FIGURE B-1. Inheritance and defaulting flow chart

Notice, for a charlist in an e-i-c (as opposed to an envdesc), both the inherit and envname attributes are relevant. The charlist's inherit value takes effect only for inheritable categories with no explicit setting for the inheritance value. Any characteristic in an explicitly mentioned category that is neither explicitly specified nor inherited (either because it is not inheritable or because its category's inherit attribute is "off") would derive its value from the default environment. Figure 1 helps to explain the movement through the inheritance and defaulting procedure. The following are some of the uninheritable categories that may be omitted from the charlist: chgmark, prespace, postspace, textbrk, and keeps. When omitted from the charlist, these rules shall be used:

a. When chgmark is omitted from the charlist, the current status of the change marking continues (see B.4.4.10).
b. When prespace or postspace is omitted from a charlist, this e-i-c contributes no prespace or postspace (see also B.4.4.11 and B.4.4.12).
c. When textbrk is omitted from a charlist, text continues to flow in the current pageset. Start line, end line, start column, and start page are considered to be "off".
d. When keeps is omitted from the charlist, the current status of the keep, scope, widow count, orphan count, and "keep floats out" characteristics remain unchanged; the keep next and keep previous are considered to be "off".

B.3.8 ORDER OF PROCESSING. Though there are generally characteristics associated with e-i-cs in a time-independent manner, the order of processing of certain constructs is significant. In particular, since multiple Savetext Construction Rules and Usetext Sources can refer to counters and other saved text, the order of processing is significant. Furthermore, since Savetext, Usetext, Reset, and Enumerate characteristics can occur within the current element's contents (potentially changing values referenced in other Savetext, Usetext, and Enumerate characteristics), whether a Savetext or Usetext is processed before or after the content of the element gets processed is also significant. For any e-i-c instance, the order of processing should be as follows:

a. The charsubsets referenced by the charlist charsubsetref attribute are merged in the order they are listed (left to right). The explicit main charlist is then merged with the result, and inheritance and defaulting occur to form the resolved charlist. Finally, the attribute specifications (whose specval/fillval conditions are satisfied) are merged into the resolved charlist (see B.3.6.5.2.4).
b. The resulting charlist is now processed sequentially, except for any Savetext, Ruling, Puttext, Putgraph, or Usetext category entries whose placement characteristic is "after". Note that the value of any variables (either Counter IDs or Savetext textids) that appear in the Savetext Conrule or Usetext Source is resolved at the time that Savetext or Usetext is processed. Any pseudo-elements that appear in the Savetext Conrule are processed (including the complete processing of the charlist associated with that pseudo-element) when the Savetext textid is processed in a Usetext source. A Usetext Source that contains several variables and

pseudo-elements is processed as if the complex Source is saved via a Savetext to a virtual textid, and then the virtual textid is immediately used in this Usetext. That is, any pseudo-elements that appear in the Usetext Source are processed in order immediately after all variables in that Usetext are resolved.

c. After the contents of the e-i-c is processed, all Savetext, Ruling, Puttext, Putgraph, or Usetext category entries whose placement characteristic is "after" (including their subchars) are processed in the order they were specified in the charlist.

d. Processing of a category that contains a subchars begins by processing the content of the subchars, followed by the processing for the category itself. The rules for the processing order within a subchars are the same as those for processing a charlist.

B.3.9 INLINE VERSUS BLOCK STRUCTURES. The characteristics defined in the OS are not all equally relevant to every element in an input document. In particular, some characteristics are inherently relevant only for elements that generate paragraph-like or "block" structures in the output stream, and should have no effect for elements that generate "inline" structures. For example, "First line indent" has no relevance for an inline structure such as an `emphasis` element's e-i-c might generate, even if, for example, the e-i-c for emphasis inherits some particular value for "First line indent" from its parent `para` element's e-i-c. Furthermore, a block structure can have only one effective value for such categories as leading (unless the "force" characteristic is on), quadding, and indent for its entire scope.

Inline and block structures are concepts that are realized only during the actual composition process. Given the independence of the start line (startln) and end line (endln) characteristics of the textbreak category, block-structuring of the final output does not have to follow the structure and hierarchy of either the input or output tree. The determination of the structures depends on the charlists in effect for the various elements, pseudo-elements, and PI e-i-cs in the output tree. In particular, an e-i-c whose effective values (that is, resolved logical value, whether specified explicitly, implicitly, or defaulted) for the textbreak characteristics of startln and endln are both "off" produces an inline structure. (If an e-i-c starts a new column or page, its startln characteristic is considered to be "on" implicitly.) An e-i-c whose effective value for either startln or endln is "on" is a block-producing e-i-c. A block-producing e-i-c results in one or two block boundaries as follows: if the effective value of its startln characteristic is "on," the beginning of its contents (just prior to its first character or other printable item) is a block boundary; if the effective value of its endln characteristic is "on," the end of its contents (just following its last character or other printable item) is a block boundary. The block boundaries divide all content in the final output stream into blocks as follows: first all sequences of consecutive block boundaries (that is, with no intervening printable items such as characters or graphics or rules in the output stream) are collapsed into a single block boundary, then the remaining block boundaries divide the final output stream into block structures.

Note that the block structures as defined are not hierarchical, that is, they do not nest as structures in the output tree do. However, the determination of what formatting

characteristics take effect for the duration of a given block structure is made with regard to the hierarchy of the e-i-cs in the output tree. Generally, all (character and other inline printable) content of the block take on formatting characteristics from e-i-c's in the output tree as follows:

a. For categories that are relevant to inline structures, use the characteristics from the e-i-c of the content's parent element (including pseudo-elements and paired OS processing instructions).
b. For the categories that are irrelevant for inline structures, look "upward" in the output tree from the content, and use the characteristics of the e-i-c of its closest ancestor element (including pseudo-elements and paired OS processing instructions) that is a block-producing e-i-c.

The indent, quadding, prespace, postspace, vjinfo, and textbreak categories are irrelevant for inline structures. The leading category is ignored for inline structures unless the force characteristic is set "on." The only characteristic in the keeps category that is relevant for inline structures is "keep". The span category implies a value of "on" for startln and therefore creates a block-producing e-i-c; most characteristics in the textbreak category also cause the e-i-c to be block-producing.

The prespace and postspace for all block structures contribute to the actual pre/postspace according to the rules for combining prespace and postspace described in B.4.4.11.

The effective values of the indent, quadding, and leading categories' characteristics are fixed for the duration of a given block structure (with the exception of leading's lead characteristic in the case that its force characteristic is "on").

The firstln indent for a block is determined by the firstln indent in effect for the first character (or other printable item) of that block. The firstln indent is used only once per output tree e-i-c. If a block-producing e-i-c has another block-producing e-i-c nested within it in the output tree, all lines in the block structure that result from the part of the outer block-producing e-i-c following the inner block-producing e-i-c shall use the left indent (rather than the firstln indent) derived from the outer block-producing e-i-c.

The leading category's lead characteristic (when force is not "on"), indent's left and right indent, and quadding's quad and last quad are all determined by the value in effect for the last character (or other printable item) of that block. While these values are commonly the same as those in effect for the first character of the block, the important differences occur in the cases of "run-in heads" which are titles that get composed as part of the paragraph. In these cases, the values of these characteristics that were specified for the title are usually ignored and the whole paragraph (including the title text) is composed using the values specified for the paragraph text.

B.3.10 SECURITY. Classified documents require that an indication of the highest level of security found on a page or sheet shall appear in the running header or footer. Each element in the source document may have a security attribute, in which the security level for its content is indicated. The Security Description is used to specify the precedence of

these security values and the strings that should be automatically generated to indicate the classification level. Security text is then identified in the header and footer areas, and its value is computed according to the values of the security attributes of the content that appears within the scope indicated. The scope can be a page, sheet, or the entire document.

The security description (secdesc) and mark description (markdesc) are used to provide page markings for classified information. Secdesc provides the capability to specify placement of normal classification markings on a page, or to place the optional document level classification on each page. Markdesc provides the capability to specify placing additional markings on the page. Such markings may include such entries as special access required, restricted data notices, or intelligence source markings. Use of these two categories should prove adequate for most security marking requirements.

Using these capabilities does not ensure that the processing system will properly mark the information, or that the information is properly protected. MIL-STD-1840 provides an interface standard for secure transmission of classified data. Protection and transmission of classified data shall be as specified in the contract or applicable DoD or agency security directives.

B.3.11 COLOR. The use of color in formatting can be specified in several areas of the Output Specification, including highlighting text, graphics, and rules. Specification of color is intended for use with “spot color” techniques; that is, a single color is used for any particular text character, graphic, rule, or background with no “merging” of colors. The color is to be used full strength except when screens are specified, in which case a percentage of full strength should be used. Within the OS, the following generic colors are allowed: black, white, red, orange, yellow, green, blue, violet, brown, and gray. It is up to the formatting system to determine the exact color used for printing or display.

B.3.12 BORDERS. A border is a graphic that underlays the page, and whose presence on any particular page is triggered by the presence of an e-i-c whose content is also placed on that page and whose charlist includes the border category.

A specific border graphic is associated with a particular logical name (also called its border indicator) via the “bordspec” element that occurs in one of the page type specifications in the page description section of a FOSI. The bordspec element’s bordent attribute specifies the graphic while its bordname attribute specifies the logical name. Then any e-i-c that should trigger a border would use the border category’s bordname characteristic to specify the logical name of the required border. Note that a recto page’s bordspec may associate a different graphic to a given logical name than the verso page’s bordspec. This allows for different graphics, each appropriately tailored to either a recto or verso page, to share the same logical name so that the triggering e-i-c can refer to one logical name regardless of the target page type.

When multiple borders are triggered for a single page, the value of the bordspec’s precedence attribute determines the logical order in which the borders will be placed on the output page. The border with the lowest precedence value will be placed “first” on the page (that is, it will underlay all other borders), followed by the border with the next

lowest precedence and so on. All borders underlay all other text and graphics placed on the page by the composition system.

B.3.13 FLOATING CONSTRUCTS. The general composition model of almost all document processing systems is sequential: the document is processed from front to back (in document order) and the input content is composed onto the output pages so that the information is presented basically in the same order in which it was input.

There are some notable exceptions to this paradigm; among them are various kinds of "floating" blocks of material, cross-references, content replicated in tables of contents, indexes, or similar constructs, text from the input content placed into running headers or footers. In the Output Specification, the Savetext and Usetext constructs can be used to handle most of these needs with the exception of floating blocks of material. Footnote processing is one case of floating that has special handling in the Output Specification. The Output Specification has a generalized mechanism to handle all other floats.

This mechanism allows for any element-in-context (e-i-c) to specify that its contents should float to some other place in the output instance than the next available location in the flowing text area. The mechanism allows for three different floating behaviors:

a. Floating material into an area on the current or nearby page at the top or bottom of a column, flowtext area, or page.
b. Floating material into an area at the top or bottom of the current column or page in such a fashion as to provide a mechanism for repeating floating elements across page breaks (for example, repeated captions).
c. Floating material in a fashion similar to that of footnotes in such a fashion that multiple references to identical floats on the same page result in only one copy of the float to appear in the floating location on the page (as might be the case with footnotes within tables or trademark and corresponding legends).

Specifying floating involves three steps:

a. Defining the characteristics of the floating layout area in the resource description.
b. Attaching a floating layout area to a page model in the page description.
c. Specifying for each e-i-c to be floated the floating layout area into which it is to be floated.

The float specifications in the Output Specification provide the output system with various constraints within which to attempt to lay out the result. The constraints must be specified so as to leave the output system some flexibility, since different composition systems will necessarily use different page layout algorithms. Using the Output Specification's floating mechanism, it is possible to provide constraints that cannot be satisfied. In this case, the output system may either issue an error or attempt some resolution that does not satisfy all the constraints. The FOSI writer should avoid writing specifications that lead to improperly constrained situations.

The float category interacts with other categories in that floating determines only layout position by the normal inheriting and defaulting procedures. The formatting properties of surrounding elements are unaffected by the floated element. Pre- and postspace are applied in the normal manner and affect spacing in the float location. The postspace of the e-i-c that precedes the floating e-i-c and the prespace of the e-i-c that follows interact in the usual way of adjacent pre/postspace. A Span specification in the charlist of the e-i-c instance that is floating affects the material that floats. However, the material that does not float is not interrupted and balanced, but rather continues in the flow text area as formatted. Keeps previous and keeps next on the e-i-c instance that is floating has no meaning and is ignored. The keep characteristic of the keeps' category is relevant and indicates whether the floated material can be broken over multiple float locations if the material cannot fit all in one location or on one page.

Textbreak characteristics of startln and endln are both treated as if they were set "on" ("1"). Startcol has no meaning and is ignored. The characteristics startpg, newpgmdl, and pageid are relevant for the floating material (but have no affect on the material that does not float) as follows: if startpg="1" and newpgmdl=local, then the floating element begins on a new page and the pageset specified by the pageid is used for the duration of the floating element's content. This allows, for example, the floating of an element into a series of landscape pages. (A value of newpgmdl=global is treated as if newpgmdl=local when the e-i-c instance in question floats.)

Multipage objects can be floated, though certain specifications (such as "float to the facing page") will lead to improperly constrained situations. When the multipage floated object is placed into the output stream by the output system, the multipage object will be placed on subsequent pages according to the output system's page breaking and float placement algorithms.

B.3.14 SIGNIFICANT RECORD ENDS. Although the SGML standard (ISO 8879) carefully defines which record ends ("carriage returns" or "line breaks") in the input source file are significant and which are to be ignored by the parser, it does not prescribe what the application should do with significant record ends.

Formatting applications in compliance with this specification should normally treat a significant record end as a space character. Consecutive multi-space characters (including record ends being treated as space characters) should normally be treated, for the purpose of composition, as a single space character. The exception is in the case of an e-i-c with "asis" quadding. In this case, each space affects formatting and each significant record end causes a line break during formatting (consecutive record ends should produce multiple line breaks, that is, they should produce a "blank line" in the composed output). In "asis" mode, it is an input error if the content given between two record ends will not fit on one output line. What happens in this case is left to the output system.

A "verbatim" environment (see for example the verbatim element in the Example DTD of MIL-HDBK-28001) is sometimes defined to format computer examples or similar text in a "typewritten emulation" mode. To get the desired formatting effect, the associated e-i-c should not only specify "asis" quadding, but may also need to cause the use of

a monospaced font as well as ensuring that the word spacing and letter spacing are set appropriately. Since in such a case only character data is usually desired, the “verbatim” element is usually defined with declared content of RCDATA. General entity references and character references are resolved in RCDATA content. Since start tags are not recognized in RCDATA content, there can be no element content.

B.3.15 MERGING OF FOSIS AND MODULARITY. Since description areas of the Output Specification (for example, rsrcdesc, secdesc, or pagedesc) are repeatable, “FOSI modules” (ospackages) may be created separately and merged together to create one FOSI. When ospackages are used, the initial set of description areas are not contained within an ospackage; this initial set is known as the “global” module.

To facilitate the ability to work on modules independently of one another, a FOSI processing system that handles multiple modules will treat the names for FOSI variables (for both strings and counters), charfills, hyphrules, floatlocs, pagesets, pagespecs, bordnames, named environments (including graphic environments), and charsubsets referenced in one module as distinct from the names in another module (that is, the modules have different name spaces for these objects). Specifically, when a name of a such an object is used in a given module, first the current module’s resource description (in the case of a FOSI string variable, counter variable, hyphrule, floatloc, or charfill), page description (in the case of pagesets, pagespecs, and bordnames), style description (in the case of named environments and charsubsets), or graph description (in the case of graphic environments) is searched for a defining specification for the name in question. If no such specification is found, then for string variables, an implied time dependent declaration is assumed for the current module. For all other objects, the appropriate section of the global module is next searched for a defining specification. If it is desired for a reference within a non-global module to refer to the object defined by a specification in the global module, the reference in the specification in the non-global module should preface the object name with the string “global:”. In this case—even if there is also an object by the same base name existing in the current module—this “global reference” will refer to the object of that name in the global module (and it would be an error if no such object existed in the global module unless the object were a string variable in which case an implied time-dependent string declaration is assumed for the global module).

Names of e-i-cs (including pseudo-elements and PIs) are always global. Best match calculation for e-i-c’s occurs over the entire FOSI without regard for its modularization with one important exception: while it is the first of a set of identical matches that is used as the matching e-i-c within any single module, it is the last module’s match that shall be used as the matching e-i-c in the case of identical matches. This allows “delta” or “override” modules appended to an existing FOSI to provide alternative e-i-c specifications for a given element-in-context. Note that this overriding occurs only if the best match algorithm determines that the final candidate set contains multiple e-i-c’s that are identical in terms of gi, context, occurrence, and gitype.

B.3.16 FOSI MANAGEMENT TECHNIQUES. Since a FOSI is an SGML document instance, a FOSI writer may take advantage of various SGML constructs in creating and

maintaining a FOSI. For example, judicious use of internal and external SGML entities may facilitate configuration management of FOSI fragments and reuse of FOSI code. With marked sections, several variant formats, such as different page sizes or paper versus screen output, can be effectively coded in the same FOSI such that appropriate settings of the various marked section keyword parameters would allow the FOSI to remain parseable while choosing from among the target variants.

B.4 KEY TO CHARACTERISTICS

In the following subsections each category or other logical group of characteristics is described in detail, according to the following structure:

a. Category name.
b. Category definition.
c. Table of characteristics and types of values allowed.
d. Explanation notes and points of clarification.

Definitions are provided for characteristics where appropriate, and restrictions on values are noted where applicable. Although this section contains some examples, it is meant to be used as a reference section, and not as a guide or tutorial. Further clarification on the actual application of characteristics in a FOSI may be obtained by studying the Output Specification DTD in B.5.

B.4.1 RESOURCE DESCRIPTION CHARACTERISTICS.

B.4.1.1 Hyphenation rules. The process of breaking a word at the end of a line of text for purposes of line justification.

Characteristics - Definitions:	Values - Definitions:
Language	ID
Dictionary	Pointer
Word Break Exceptions - A specification for preferred hyphenation points.	Pointer
Unbreakable Words A specification for words with no allowable hyphenation points.	Pointer
Break Characters - Characters which can be broken before or after without inserting a hyphenation character.	String
Break Before Characters - Characters which can be broken before without inserting a hyphenation character.	String
Break After Characters - Characters which can be broken after without inserting a hyphenation character.	String
No-break characters - Characters before or after which it is not legal to break.	String
Hyphenation Type - Specification of the method for making hyphenation decisions.	List (Dictionary, Logic, Both, Any)
Hyphenation Zone - Specifies the amount of space from the margin that can be left by choosing not to hyphenate (aesthetic rag).	Size/Distance

Ladder - The number of consecutive lines in a column that can end with a hyphenation.	Integer
Minimum characters left on a line.	Integer
Minimum characters pushed to the next line.	Integer
Hyphenation across columns allowed.	Toggle
Hyphenation across pages allowed.	Toggle

EXPLANATION:

a. Using the hyphrule's language attribute and the hyphen category's "lang" characteristic, any e-i-c can refer to a particular one of the multiple hyphrules thereby allowing change of hyphenation schemes within a document. (A given implementation may restrict the granularity at which hyphenation schemes can be changed.)

b. The Dictionary attribute (hjdata) allows specification of a (non-exception) dictionary, since with multiple hyphrules, there can be multiple (non-exception) dictionaries. Exactly how this dictionary is accessed and what its format may be depends on the value of the Hyphenation Type characteristic and the output system.

c. Discretionary hyphens are accommodated through the Word Break Exception List.

d. Word Break Exceptions always take precedence.

e. The syntax to be used for Word Break Exceptions requires each word to be separated by a comma or a space, with each hyphenation point marked by a hyphen, for example, "ref-er-ence, syn-tax". The use of double hyphens indicate a fixed hyphen, for example, MIL–PRF–28001.

f. The syntax to be used for Unbreakable Words requires each word to be separated by a comma and an optional space.

g. Break Characters, Break Before Characters, and Break After Characters only have relevance if hyphenation is allowed. They do not provide possible blind breaks if hyphenation is disallowed.

h. The Hyphenation Type specifies the general method that the composition system should use to make hyphenation decisions. "Dictionary" specifies that all hyphenation decisions occur based on lookup in some dictionary; however, dictionaries are not interchanged because they may be proprietary. A specific dictionary may be specified per contract. Likewise, "Logic" specifies an algorithmic approach, but no algorithms are interchanged. "Both" specifies a combination of the two approaches, and "Any" specifies that any approach is allowed.

i. For an explanation of hyphenation zone see B.4.4.3.

B.4.1.2 Character fill. Describes a literal used to fill a space horizontally or vertically.

Characteristics - Definitions:	Values - Definitions:
Character Fill literal - The character string.	String
Orientation - The direction of the fill.	List (Vertical - fill is down from baseline. Horizontal - fill is to the right from cursor).
Type	List (RR - Ragged left, Ragged right; RF - Ragged left, Flush right; FF - Flush left, Flush right; FR - Flush left, Ragged right)
Space Before - Minimum spacing before the first occurrence of the specified literal in the fill.	Size/Distance
Space After - Spacing after the last occurrence of the specified literal in the fill.	Size/Distance
Padding - Spacing between occurrences of the specified literal.	Size/Distance
Truncation - Whether the last copy of the literal string can be truncated.	Toggle
Suppression Rules - Do not apply fill if less than this many copies of the literal string would appear.	Integer
Alignment - Vertical alignment of the fill literal.	Toggle
Character Fill ID - a unique name for the character fill.	ID
Minimum Count - Minimum number of the copies of the literal to fill	Integer
Break - Determines where line breaks are permitted	List (None, Before, Within, After, BeforeWithin, WithinAfter, BeforeAfter, Any)

EXPLANATION:

a. The literal specifies the character (or string of characters) that will be repeated some number of times (depending partly on other character fill characteristics) to fill the available space in the current constrained area in the output. It is allowable, and often useful, for the literal to be a space.

b. The Type characteristic determines where the literal begins and ends. “Ragged” specifies that the literal begins or ends immediately after or before the text it

abuts. “Flush” specifies that all literals begin or end at a “margin” determined by the longest text before or after the literal.

c. When the Alignment characteristic has a value of “on”, the Literal shall align vertically on successive lines. When the value is “off”, the Literal shall immediately succeed the previous character on each line.

d. Specifying Character Fill only sets up the characteristics for a character fill string. To specify the position of the string in the output, the Character Fill ID must be used in the Source of a Usetext specification (either directly or through its inclusion in a Construction Rule of a Savetext). If more than one character fill is positioned within the same horizontally constrained area of a line in the output, each character fill in that area will fill the same amount. It is possible to constrain an area so that a character fill must “back up” horizontally in the output, that is, it must act as a “negative space.” This is permitted (assuming neither Suppression Rules nor Minimum Count characteristics prohibit it), but any value of the Character Fill literal is ignored and the effect is as if a blank literal were specified.

e. If Minimum count is greater than or equal to the Suppression Rule value, then the Minimum count determines the number of copies of the literal to be used even if this forces a line break. If Minimum count is less than the Suppression Rule value, no minimum fill number is forced.

f. The Break characteristic determines where line breaks are permitted with respect to the charfill. If Break is After, then any necessary line break should occur immediately after the charfill. If Break is Before, then any necessary line break should occur immediately before the charfill. If Break is Within, then any necessary line break should occur within the charfill subject to the constraints put on it by the Minimum count characteristic. If Break is any of BeforeAfter, BeforeWithin, or WithinAfter, then line breaks are allowed in either of the two locations indicated by the characteristic value’s name. If Break is None, then the charfill is unbreakable and breaks are disallowed both immediately before and after it (though if this over-constrains the composition process, the results are output system dependent). If Break is Any, then a break may occur anywhere before, within, or after the charfill.

B.4.1.3 Counter. This construct is used to specify the properties of a counter that will be associated with one or more e-i-cs using the Enumerate category.

Characteristics - Definitions:	Values - Definitions:
Initial - The initialized value of a counter.	Integer
Style - The style in which the counter is to be displayed.	List (Arabic, Roman Upper, Roman Lower, Alpha Upper, Alpha Lower, User Defined)

Specified Style - A user-defined list of characters or symbols in order of precedence, for example, "* ** # ## ...".	String
Sequence - Used to describe the sequence for alphabetic display of counters exceeding 26.	List (1, 2)
Padding length - Used to describe the minimum number of characters in the string resulting from the conversion of a counter into arabic style.	Integer
Exceptions - Used to list any exceptions to the Style or Sequence. For instance, if the style is Alpha Upper, "I" and "O" might be exceptions.	String
Counter ID - A unique name assigned to a counter.	ID

EXPLANATION:

a. When the Style characteristic is “User Defined”, the value of the Specified Style is used to determine the values for the enumeration counter. When the Style characteristic has any other value, the Specified Style is ignored.

b. The Sequence characteristic does not apply when the value of the Style characteristic is “Arabic”, “Roman Upper”, “Roman Lower”, or “User Defined”. The result is either upper case when the Style characteristic is “upper”, or lower case when the Style characteristic is “lower”. Following are the sequence rules for enumeration (counters):

   1. Following “Z” continue enumeration with: AA, AB, AC, ... AZ, BA, BB, BC, ... BZ, ..., ZA, ... ZZ.
   2. Following “Z” continue enumeration with: AA, BB, CC, ... ZZ, AAA, BBB, CCC, ... ZZZ.

c. Compound numbers are specified through the Construction Rule of the Savetext characteristic.

d. A counter value of 1 shall correspond to the first symbol in the set of characters implied by the Style or determined by the Sequence. (For example, a value of 3 would correspond to III, iii, C, and c for Styles of Roman Upper, Roman Lower, Alpha Upper, and Alpha Lower respectively.) When the Style is other than Arabic, what to do with a counter value of zero is left to be resolved by the output system.

e. The padding length characteristic is a positive integer giving the desired final length in characters of the string that gets placed into the output stream whenever this counter is used in a Usetext Source (or into the Savetext textid variable when used in the Savetext Construction Rule). This length is achieved by padding the character representation of the current value of the counter on the left with zeros. The padding length is ignored except for a Style specification of Arabic (either in the counter declaration or via a local overriding style specification in the Construction Rule or Source).

B.4.1.4 String. This construct is used to specify the properties of a string that will be associated with one or more e-i-cs using the savetext or usetext category.

Characteristics - Definitions:	Values - Definitions:
Savetext Name - A unique name for the saved text.	NMTOKEN
Literal - The initial string value.	String
Time Status - Indicates whether this variable is time dependent (0) or time independent (1).	Toggle
Scope	NAMES
Export - Indicates whether the Savetext Name’s saved text will be “exportable”	Toggle

EXPLANATION:

a. The Savetext Name follows the same rules as that for the Savetext category; the Literal follows the same rules as that for the Puttext category.

b. A Time Status of zero (disabled) indicates that the variable named by the Savetext Name is time dependent. That is, the value of the variable will be evaluated at the time it is referenced in a Savetext Construction Rule or Usetext Source. A Time Status of one (enabled) indicates that the variable is time independent; that is, the value that is used to replace this Savetext Name everywhere this variable appears is the value this variable would have before the end of its scope. All time independent variables must be declared using the String construct; any variable not declared will be time dependent with one exception: any savetext textid variable whose name is filled in from a source document attribute is time-independent.

c. In the case of time independent variables, the contents of the variable named by the Savetext Name will be resolved. That is, its time independent value will be determined after completion of the processing of the element instance listed in the Scope characteristic whose scope has already started and whose end tag

is first encountered after the e-i-c instance in which the reference to this time independent variable occurs. If any element in the variable's scope is a descendant of another element in the scope, the result is left to be determined by the output system. If no element specified by the Scope characteristic is an ancestor of the e-i-c instance in which the reference to this time independent variable occurs, the variable will be resolved at the end of the document element.

d. If the Export toggle characteristic is left #IMPLIED, it is treated as "off." When this toggle is on, the associated savetext name will name an "exportable" variable. The meaning of the rest of the attributes (for example, literal, status, scope) remain as with non-exportable variables. (Note in particular that one probably wants to be using "append=1" on savetexts done to this variable if one wishes to accumulate the results of multiple savetexts to this textid. Furthermore, in cases such as tables of contents, one probably wants to make the variable time independent.)

e. An exportable variable is one for which the composition system must do whatever is necessary to instantiate on the local file system the material that is saved in this variable so that external processing is possible. The syntax and semantics of savetext's conrule and usetext's source remain effectively unchanged. What is logically getting exported is neither literal text nor SGML, but rather a Construction Rule. In particular, note that literal text chunks (the characters delimited by backslashes) are basically RCDATA chunks, and characters that could otherwise potentially be mistaken for markup characters would be treated in such a way (for example, as character entity references) that they would not be confused with markup. Note that this is consistent with the way conrules are currently treated in the non-exportable case. Therefore, the only way to export material that is to be represented in such a way that it is later treated as markup is via a pseudo-element in a conrule or via #CONTENT.

f. Spacing and padding tokens of a construction rule are not instantiated in an exported variable's value; rather such tokens are stripped out of the exported material.

g. To indicate a record end in exported material, a "&#RE;" character entity reference (which would be specified in a literal string token of a Construction Rule) shall be used.

B.4.1.5 Floating locations. A float location (floatloc) is similar to a layout area in the page model. Elements float into a float location at the time the float category (in a charlist or subchars or charsubset) is encountered. Floats in a given float location are maintained in a first-in-first-out order except that certain combinations of specifications (such as a float with pagetype=next followed by one with pagetype=same) can provide constraints that contradict the first-in-first-out principle, in which case the output system may choose to violate the first-in-first-out principle as part of its fallback resolution.

APPENDIX B

Characteristics - Definitions:	Values - Definitions:
Float ID - A unique ID for this floating location.	ID
Float Type - Indicates the kind of float emission behavior.	List (Once, Repeat At Page Break, Repeat at Column Break, Multi-reference)
Maximum Depth - The maximum extent to which the sum of all areas associated with this floatloc can grow on any given output page.	Size/Distance
Minimum - The least amount of space that can appear	Size/Distance
Nominal - The preferred amount of space (what should appear)	Size/Distance
Maximum - The greatest amount of space that can appear	Size/Distance

EXPLANATION:

a. A floatid identifies the float location so that it can be referenced by the elements that are to float into it. (Float Ids are also referenced in the Page Description and possibly in the Keeps and Reset categories.)

b. The maximum depth is a dimension specifying the maximum amount for the sum of all layout areas associated with this floatloc (as specified by the floatid) for any one page.

c. A vertical spacing amount defined by minimum, nominal, and maximum separates the layout area from the (column or flowing text) text area or from the adjacent float layout area. That is, for each float location instance on a given output page, any pre-/postspace on the “text-side” of the float location area (for example, initial space for bottom floats and final space for top floats) is ignored and replaced by this vertical spacing amount. Furthermore, any initial prespace of the first float location at the top of the page and any final postspace of the last float location at the bottom of the page are ignored so that the outermost float abuts the next outer layout area on the page (for example, the “Space Above” area in the case of the first float location at the top of the flowing text area). Between individual floats within a single float location, the pre-/postspace specified in each floated element combines in the usual way with the pre-/postspace of adjacent floats.

d. The output system will place floats associated with a given float location onto output pages in first-in-first-out order within the various constraints of page location, page type, float and float location width, and maximum depths. Where

it is not possible to satisfy all constraints, the output system can issue an error message or employ some fallback algorithm.

e. Most floats (whose Float Type is Once) appear once and only once for each reference in the flowtext. However, if Float Type is Repeat At Page Break, the material in this float location will be repeated at each page break until the scope of this floated material ends. Finally, if Float Type is Multi-reference, the float appears exactly once per page for any page from which this float was referenced one or more times.

B.4.2 SECURITY DESCRIPTION CHARACTERISTICS. The security description (secdesc) specifies the string variables for which the composition system is requested to perform a certain kind of special processing. In particular, each variable has an ordered set of possible values associated with it. The highest value ever assigned to the variable within a particular composed scope (for example, page, sheet, or document) is the value to which it should resolve when reference is made to this variable from the header or footer of the page description. Such special processing is designed to be appropriate for most security marking generation requirements.

The OS provides a mostly automatic way to handle basic security markings, plus a more elaborate way to handle more involved markings. The simple method—called the “sectoken” method—requires the specification of the name and possible values (and relative priority order) of the attribute in the source DTD that is used to assign security levels to source elements. The FOSI provides this information as the value of the two characteristics on the secdesc category. Next, the FOSI must specify what string should be generated for the security text for each possible value of the source DTD’s security attribute (for example, generate “SECRET” for an attribute value of “s”). The security token (sectoken) category’s characteristics specify this mapping. When the security description is so set up, the composition system automatically computes the highest level of security (as indicated by the use of the security level attribute on the source elements) occurring on each page and places the appropriate corresponding string into the output stream in place of the sectext element occurrence in the header/footer as specified in the page description (pagedesc). This method requires no explicit string variable references; instead, the composition system maintains internal equivalents of string variables whose contents are automatically maintained and used.

The OS also provides a security description processing mechanism somewhat analogous to attribute specification processing (the attspec using the att element) in an e-i-c. While the FOSI must still provide the name and possible values of the relevant attribute in the source DTD, a security attribute (secatt) element can specify processing similar to that provided by e-i-c’s attspec specval and fillval processing. The description of the markval, markfill, markover, and marktext elements provide greater detail. This method is known as the “secatt” method.

B.4.2.1 Security description (secdesc). Defines the security levels and the priority order of the levels. Proper specification of the attribute specification and security order characteristics is necessary for both the sectoken and secatt methods of security description processing.

The secdesc element contains occurrences of sectoken elements (for sectoken method processing) or secatt elements (for secatt method processing). Note that the two methods of processing can be mixed in a given secdesc.

Characteristics - Definitions:	Values - Definitions:
Attribute specification	String
Security order	Name tokens
Ignore security	String
Security description ID	ID

EXPLANATION:

a. The attribute specification characteristic identifies the attribute in the source document used to indicate the security levels. The attribute name whose value gives the security level of an object must be the same on all elements in the source DTD.
b. The security order characteristic is used to list the priority order of the security levels in descending order, for example, “s c u”, where “s” indicates the highest level and “u” indicates the lowest level.
c. When security considerations are not applicable, the Ignore Security characteristic indicates the value of the security attribute of the document element that indicates that security processing should not be done. In this case, security tokens specified in headers and footers will not be replaced with any security text. For example, when Ignore Security is set to “u” and the security attribute of the document element is set to “u” (for example, <doc security="u">), no security text appears in the headers or footers. Note that this characteristic is only relevant to the sectoken method of processing since the secatt method includes the facility to test document element security attributes directly.
d. The security description ID should be used when more than one security description is provided in a given FOSI so that subsequent occurrences of the sectext element in the page description can indicate (via sectext’s secref attribute) the desired security description.

B.4.2.2 Security token (sectoken). Defines the security markings that appear in the header and footer as specified by the sectext category. For each value specified in the security order characteristic of the secdesc category, a corresponding Security Text string should be specified. Each occurrence of the sectoken category associates one security order characteristic value (via the sectoken’s security value characteristic) with one Security Text string.

When a security text (sectext) category occurs in a header or footer, the highest level of security occurring within the specified scope (page, sheet, or document) is computed,

and the Security Text associated with that value is placed into the output stream in place of the sectext category. If no Security Text is associated with a security level, no text appears. [Do not confuse the Security Text (sectext) characteristic of the Security Token (sectoken) element which is part of the Security Description (secdesc) with the Security Text (sectext) element which is allowed within headers and footers within the Page Description (pagedesc).]

The sectoken element is used for specific security processing using the sectoken method. No other element in the security description is used for the sectoken method. The necessary testing, storing, and processing is performed by the composition system using implicit "variables" that are only referenceable via the sectext element in a header or footer specification in the page description (see B.4.3.2.11.2).

Use of the sectoken element may be necessary for processing using the secatt method. In particular, most uses of the markfill element expect that occurrences of the sectoken element have set up correspondences between security value tokens and security text strings.

Characteristics - Definitions:	Values - Definitions:
Security value	Name token
Security text	String

EXPLANATION:

a. When a security text (sectext) element occurs in a header or footer, the highest level of security occurring on the page is computed, and the Security Text associated with that value is placed into the output stream in place of the "sectext" element. If no Security Text is associated with a security level, no text appears.

b. If no Security Text is associated with a particular security value, the null string is assumed at the security text to be associated with this security value.

B.4.2.3 Security attribute specification (secatt). Provides an alternate method to specifying the security markings that appear in the header and footer. The security attribute specification (secatt) provides the FOSI writer with additional flexibility to place security and security related markings from the source document. It has a logic attribute which allows the user to specify formatting based on a combination of security events. The functionality of the security attribute (secatt) roughly parallels that of the attribute specification (attspec) construct subordinate to an element-in-context.

The secatt element may contain combinations of the markval, markfill, and markover elements, all of which are "tests" that may succeed or fail depending on values of attributes on elements in the source document. (By rough analogy, markval is similar to specval while markfill and markover have similarities to fillval.) There may be multiple secatts in a secdesc. An entire secatt will be processed only if the combination of all its tests succeeds. The logic attribute determines whether the combination succeeds if at least

one of the markval, markfill, or markover tests succeeds (logic="or") or only if all of the markval, markfill, and markover tests succeed (logic="and").

If the secatt is processed, the (possibly implicit) marktext will be processed resulting in the assignment of some value to some variable that can later be referenced in the header or footer of a page description. See the individual descriptions of markval, markfill, markover, and marktext for more information.

A secatt may contain zero, one, or more markvals, zero or one markfill, and zero or one markover. In particular, it may contain no markvals, markfills, or markovers at all, in which case the secatt test always succeeds and the marktext will be processed. All secatt's are considered to have exactly one marktext. If there is no explicit marktext, the secatt is effectively considered to be equivalent to one containing a marktext element with no assigned attributes. (See the description of marktext for the defaults for the various marktext attributes.)

Characteristics - Definitions:	Values - Definitions:
Logic	List (and, or)

EXPLANATION

a. If the logic attribute is "or", then the combination succeeds and the secatt is processed if at least one of the markval, markfill, or markover tests succeeds. If the logic attribute is "and", then the combination succeeds only if all of the markval, markfill, and markover tests succeed. The default is logic="and".

B.4.2.3.1 Markval. The markval element in a security attribute specification is somewhat analogous to the specval element in an e-i-c's att specification. It specifies a test that either succeeds or fails, and the result of this test (along with the results of the other tests in this secatt combined as specified by the value of the logic attribute on this secatt) determines whether the (possibly implicit) marktext in this secatt is processed.

The test specified by this markval is to compare the highest value of security (as specified by the Attribute specification and Security order attributes on this Security description's secdesc element) within the current page, sheet, or document (as specified by this markval's Scope attribute) against a specific string value specified by this markval's Attribute value (attval) characteristic.

Characteristics - Definitions:	Values - Definitions:
Scope	List (document, sheet, page)
Attribute value	String

EXPLANATION

a. If Scope is omitted, page is assumed.
b. Document-wide security computation does not apply to any structures in a subdoc external entity. The results of use of document-wide security with subdocs is output system dependent.
c. This markval succeeds only if the attribute value (attval) string equals the highest value of security within the specified scope—that is, it must be one of the name tokens listed in this secdesc's secorder characteristic. The markval test is case-insensitive.

B.4.2.3.2 Markfill. The markfill element in a security attribute specification is somewhat analogous to the fillval element in an e-i-c's att specification. Its primary role is to specify a scope from which the highest security value therein used shall be retrieved for use in the associated marktext. Secondarily, it also specifies a test that either succeeds or fails and whose result (along with the results of the other tests in this secatt combined as specified by the value of the logic attribute on this secatt) determines whether the (possibly implicit) marktext in this secatt is processed. The only way the markfill test fails is if there has been no assignment to the security attribute (as specified by this secdesc's Attribute Specification characteristic) in any element any part of which is contained within the specified scope. (Since an element's content by definition includes all descendant elements, a markfill would fail only if, for all elements any part of which are in the specified scope, none of those elements and none of any of those element's ancestors had a value for the security attribute.) Note that a given secatt can contain at most one markfill.

Characteristics - Definitions:	Values - Definitions:
Scope	List (document, sheet, page)

EXPLANATION

a. If Scope is omitted, page is assumed.
b. Document-wide security computation does not apply to any structures in a subdoc external entity. The results of use of document-wide security with subdocs is output system dependent.
c. The highest value of the security attribute (that is, the attribute specified by this secdesc's Attribute Specification characteristic) within the specified scope is retrieved by the composition system for possible subsequent reference of its associated security text by this secatt's marktext (via #CONTENT). Note that, while the retrieved value will be one of the name tokens listed in this secdesc's secorder characteristic, the value that is made available for this secatt's marktext in #CONTENT is the security text string associated with this token as specified by the appropriate sectoken occurrence in this security description.

B.4.2.3.3 Markover. The markover element in a security attribute specification is for special case "overrides" for security. It too is somewhat analogous to the fillval element in an

e-i-c's att specification. Its primary role is to specify a source document attribute value to retrieve for use in the associated marktext. Secondarily, it also specifies a test that either succeeds or fails and whose result (along with the results of the other tests in this secatt combined as specified by the value of the logic attribute on this secatt) determines whether the (possibly implicit) marktext in this secatt is processed.

The markover element specifies (via its attloc and attname attributes which work much as the same attributes of specval and fillval) a source document attribute whose assigned value is retrieved for potential use by this secatt's marktext. If the specified attribute has no assigned value, this markover test fails. If the specified attribute has an assigned value, the test succeeds and the assigned value is retrieved by the composition system for possible subsequent reference by this secatt's marktext. Note that a given secatt can contain at most one markover.

Characteristics - Definitions:	Values - Definitions:
Attribute name - name of source document attribute being specified	Name
Attribute location - name of source document element on which the specified attribute is located	Name

EXPLANATION

a. The Attribute location (attloc) and Attribute name (attname) characteristics determine the attribute referenced by this markover—see B.3.6.5.2.3.
b. If attloc is omitted, the document element is assumed.
c. The assigned value of the attribute specified via attloc and attname is retrieved by the composition system for possible subsequent reference by this secatt's marktext (via #CONTENT).

B.4.2.3.4 Marktext. The marktext category functions somewhat as the savetext category within the charsubset of an e-i-c's att specification. It has a security construction rule (secrule) characteristic that is similar to savetext's construction rule (conrule), and it has textid and append characteristics similar to those of savetext, as well as a few other characteristics to provide special processing. A secatt's marktext is only processed if the combination of tests specified by this secatt's markval(s), markfill, and markover are satisfied.

A secatt can have at most one marktext. In effect, all secatts are considered to have exactly one marktext. If there is no explicit marktext, the secatt is effectively considered to be equivalent to one containing a marktext element with no assigned attributes.

APPENDIX B

Characteristics - Definitions:	Values - Definitions:
Text identifier	Name token
Security construction	String
Append	Toggle
Unique	Toggle
Append before - string to prepend to the value of the variable specified by text identifier	String
Append separator - string to insert between each component in the value of the variable specified by text identifier	String
Append after - string to append to the value of the variable specified by text identifier	String
Sort	Toggle

EXPLANATION

a. The text identifier (textid) specifies the name of a variable that can be used by a usetext within the header or footer of a page description. Assignment to or reference of this variable elsewhere than in this same secdesc or in the header or footer of a page description is undefined, and the results are system dependent.

b. If the text identifier (textid) characteristic is unassigned and this secatt contains a markfill whose test succeeded, then the variable used by this marktext shall be the implicit “variable” (only referenceable via the sectext element in a header or footer specification in the page description) associated with the scope specified on the markfill element. If the text identifier (textid) characteristic is unassigned and this secatt does not contain a successful markfill but it does contain a successful markval, then the variable used by this marktext shall be the implicit “variable” associated with the scope specified on the markval element. If the text identifier (textid) characteristic is unassigned and this secatt has neither a successful mark-val nor a successful markfill, then the variable used by this marktext shall be the implicit “variable” associated with the document scope.

c. The security construction rule (secrule) specifies the value to be stored (or appended) to the specified textid. A secrule follows the syntax and semantics of a savetext construction rule (conrule) with the following modification.

    1. The only conrule constructs allowed in a secrule are #CONTENT, literal text, and spacing specifications; in particular, the following are not allowed in secrules: pseudo-elements, string variables, counter names, #CONTENT(attribute-name), #XREF.

2. Furthermore, #CONTENT in a secrule is replaced by the value of the source document attribute specified by the markover element of this secatt, provided this secatt had a markover element and the markover element's test succeeded. If this secatt had no markover or its markover's test failed, then #CONTENT is replaced by the value retrieved by this secatt's markfill element, provided this secatt had a markfill element and the markfill element's test succeeded. If this secatt had neither a successful markfill nor a successful markover element, then #CONTENT shall be the null string.

d. If the security construction rule (secrule) characteristic is not assigned, this marktext will be processed as if there were a value of "#CONTENT" assigned to the secrule characteristic.

e. The Append toggle determines whether additional marking requirements will be appended to the current information or if the information will be replaced. Note that the process of "appending" may be complicated by the remainder of the marktext characteristics. The remainder of the marktext characteristics (Unique, Append before, Append separator, Append after, and Sort) have no effect and are ignored unless Append is true.

f. When appending, if Unique is true, the current value of the secrule characteristic is ignored (and nothing is appended) if that value is already part of the value of the textid variable.

g. When appending, the string specified by Append Before (if any is specified) is prepended to the value of the textid variable.

h. When appending, the string specified by Append Separator (if any is specified) is inserted between each component in the value of the textid variable.

i. When appending, the string specified by Append After (if any is specified) is appended to the value of the textid variable.

j. When appending, if the sort toggle is true, each component is inserted into the string which is the value of the textid in order. (The precise collating sequence may be system dependent.)

B.4.3 PAGE MODELS AND LAYOUT CHARACTERISTICS. This section is divided into two major subsections: Concepts, and Page Models and Pagination Characteristics. The Concepts section (B.4.3.1) describes the general concepts relating to text flow, bind margins, page model type, pageset triples, page based variable processing, and FOSI techniques with regards to specifying pagesets. The Page models and pagination characteristics section (B.4.3.2) describes the kinds of layout areas that can exist on a page, what their relationship is to each other and the characteristics for the page model categories.

B.4.3.1 Concepts.

B.4.3.1.1 Text flow. As a general rule, text flows into columns on a page, filling each column. If there is more than one column, the text flows from the top of the leftmost column to the bottom of the rightmost column before continuing on the succeeding page. Exceptions to this general rule are discussed in their respective sections, including elements

to be placed in the Header or Footer Areas, footnotes, floating elements, and elements affected by the Textbreak and Balance characteristics.

B.4.3.1.2 Multiple page models. Documents are typically made up of different types of pages. For instance, the body of a document may have pages containing two columns of flowing text, while the front matter has pages containing a single column. Documents may also contain foldout pages that have entirely different dimensions from other pages. It is therefore a requirement of a FOSI to describe a Page Model for each type of page to be generated. This specification describes general rules for describing any Page Model required for technical documents.

B.4.3.1.3 Pageset triples. The model for a page set allows for one or more “recto, verso, blank” triples (plus header/footer redefinitions for blank front/back pages). Whenever processing switches to a different page set, the first page produced using that page set will use the appropriate entry (depending if the page is a recto, verso, or blank page) from the first triple of that page set; the second page will use the appropriate entry from the second triple of that page set; and so on with the last triple in the page set being used for all subsequent pages produced. This allows, for example, for opening pages of a page set to use a different page model than subsequent ones. The most common case is usually a page set specification with only one “recto, verso, blank” triple. In the recto and verso elements, there is an optional Page Resource element in which one can include occurrences of the Reset, Enumerate, and Savetext categories. This allows counters and string variables to be managed on a per page basis. For example, one would most likely increment the counter that is being used to count pages (as all or part of the page folio) in an Enumerate in the Page Resource element of all pages.

B.4.3.1.4 Inheritance. Inheritance is generally not used for the specification of Pagination characteristics. FOSIs must usually supply values for all required characteristics of a Layout Area. For e-i-cs whose elements appear in a usetext in the header or footer areas, inheritance follows the nesting of the e-i-cs themselves, if any; the page layout area structures do not contribute to the ancestry of the e-i-cs; and an ultimate ancestral logical structure is defined that is represented in an e-i-c’s context string by the token #PAGEDESC.

B.4.3.1.5 Variable order of processing. It is possible to specify strings and counters which may be altered by the page resource, the header, footer and the e-i-cs of elements whose content appears in the flowing text of the page. The order of processing is in the following order: first the page resource and then the header and the footer.

The relative order of processing of the page description specification (for example, the pageres, header, and footer) and the e-i-cs, the results of whose contents appear within the flowing text of a given page, is not determined. In general, the processing of the flowing text contents and the e-i-cs occurs in a logically continuous fashion while the processing of the pagedesc specification that “cuts up” the result into pages happens in an “asynchronous” fashion. Therefore, to allow the pagedesc specification to refer to values of FOSI variables that are modified by e-i-c’s in the flowing text processing, there must be a mechanism to help synchronize the two disjoint processes. The TO, BO, FI modifiers described in B.3.4.10.5 provide this needed synchronization.

B.4.3.1.6 Pageset specification techniques. In the specification of multiple pagesets there will often be many similarities among the pagesets. By referencing the ID specified on the pagespec element with the pgid attribute via the pgidref attribute on the pageref element one can access these similarities to simplify FOSI writing. When this is done, the characteristics specified on the categories contained in the referenced pagespec will be merged into rectopg, versopg or blankpg being specified. To override these values or to specify characteristics which were not specified in the referenced pagespec, set values for them using the appropriate category following the pageref element. If a flowtext is defined in this way it will override only those values for a flowtext category in the referenced pagespec with the same value for the numcols characteristic. Note this same process of overriding or adding to the characteristics from the referenced pagespec applies to the elements which are children of those elements in the content model of pagespec. The most common use of referencing a pagespec is likely to be in redefining headers and footers.

B.4.3.1.7 Layout areas. Each Layout Area is comprised of Subordinate Layout Areas, unless it is a terminal Layout Area, in which case it is either empty or comprised of composition characteristics. A Page Model shall also define related printing information for paper media such as the bind edge and necessary characteristics for one- or two-sided printing. Each Layout Area is described in the following section with general rules that apply to all FOSIs. Further, rules that are specified apply to all allowable Page Models. The following outline gives the structure of the Page Model Layout Areas.

Page Set

Page Area

Top Margin Area
Bottom Margin Area
Left Margin Area
Right Margin Area
Change Mark Area
Header Area
Footer Area
Flowing Text Area

Column Area
Footnote Area
Gutter Area

B.4.3.2 Page models and pagination characteristics. This section describes general rules for creating Page Models and describes the pagination characteristics. A Page Model defines:

a. Each type of Layout Area that is allowed to occur on any page covered by the particular Page Model.

b. How each Layout Area on a page relates to other Layout Areas on the page, in particular, which Layout Areas are contained within others, and what the spatial relationships are among the Layout Areas.

B.4.3.2.1 Page set (pageset). A Page Set contains information that applies to a particular Page Area. A Page Area may cover recto pages, verso pages, recto pages with blank backs, verso pages with blank fronts, and blank pages. There can be any number of Page Sets for a document. Page Sets are referenced by associated unique ids. Figure 2 gives a graphical representation of the Page Model Layout Area.

The following characteristics apply to a Page Set.

Characteristics - Definitions:	Values - Definitions:
Page Model ID	ID
Recto/Verso	Toggle
Blank Page	Toggle
Orientation	List (Portrait, Landscape)
Media Information	String

EXPLANATION:

a. The pgid characteristics defines an unique ID for the pageset which is referenced via the pageid attribute on the textbrk category.
b. The Recto/Verso (rectver) characteristic works in conjunction with the Bind Margin characteristic defined for the recto page in the Page Set. If the Recto/Verso characteristic is turned on, the verso pages have In Margin and Out Margin Width characteristic values exactly reversed and all elements that have the Quadding characteristic set to “in” or “out” follow the Bind Edge that is currently in effect. If the Recto/Verso characteristic is turned off, the In and Out Margin retain the same values regardless of whether the page is recto or verso. (See B.4.3.2.2 for a discussion on Bind Margin.)
c. If the Page Set defines a blank page model, then when a blank page is generated (as a possible result of a Start Page characteristic value of verso or recto), the blank page model for the Page Set is used unless the Blank Page characteristic is turned off. Note that a new Page Set takes effect with the first non-blank page generated after the Page Model has changed. Therefore the Page Set that is in effect for the blank page is the one in effect before any page model change that may take place due to the Textbrk characteristics associated with the current element.
d. The standard page orientation is Portrait. A landscape page is one in which the contents of the Flowing Text and Flowing Text Change Mark areas are rotated 90 degrees counter-clockwise. The rest of the layout areas (for example the header and footer areas and right and left margins) remain in place.

e. The Media Information (mediainfo) characteristic is a string that specifies a logical name that identifies any special processing to be performed on this page (or set of pages) by the output device. Interpretation of mediainfo logical names is beyond the scope of the output specification. If mediainfo is specified for both a specific page (for example, recto) and that page's pageset, the specific page's mediainfo will take precedence.

APPENDIX B

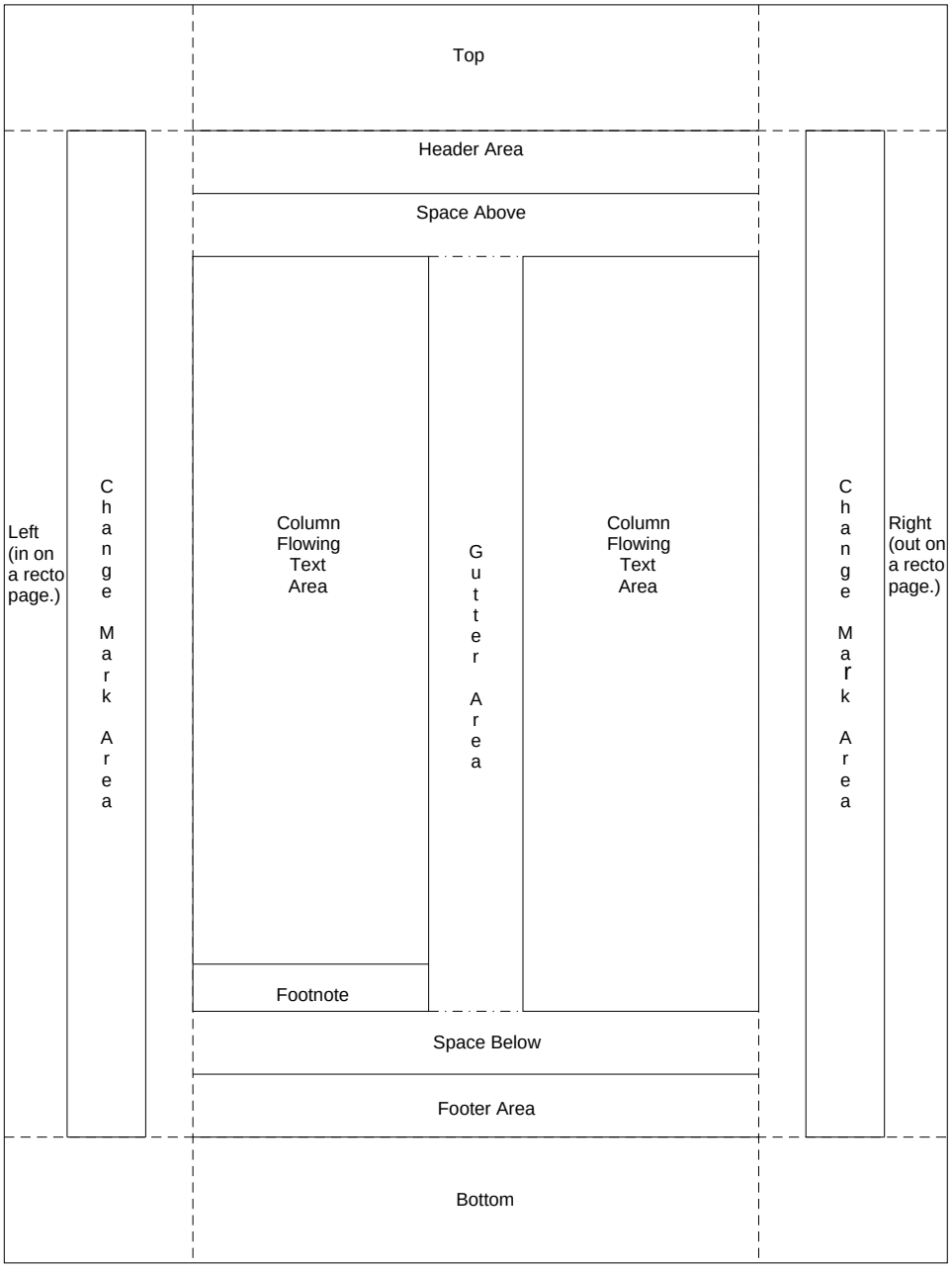

FIGURE B-2. Page model layout

B.4.3.2.2 Page area (rectopg, versopg, and blankpg). The Page Area is defined as the total physical page area. The page area is specified through the rectopg, versopg and blankpg elements. All other Layout Areas reside within the Page Area.

Content placed on a blankpg will be placed according to the rules for the placing of flowing text. If a flowtext is not explicitly set a one column flowtext area will be assumed.

Subordinate Layout Areas:

- Top Margin Area
- Bottom Margin Area
- Left Margin Area
- Right Margin Area
- Change Mark Area
- Header Area
- Footer Area
- Flowing Text Area
  - Column Area
  - Footnote Area
  - Gutter Area

Borders also appear on pages but due to their nature cannot be considered as part of the hierarchy.

These characteristics apply to a given page, that is, a given recto, verso, or blank page.

Characteristics - Definitions:	Values - Definitions:
Width	Size/Distance
Nominal Depth	Size/Distance
In (Bind) Margin	List (Left, Top, Bottom)
Change Mark Width	Size/Distance
Change Mark Offset	Size/Distance
Change Mark Placement	List (Left, Right, In, Out, Left/Right)
Top Float	IDREFS
Bottom Float	IDREFS
Media Information	String
Maximum float percent - maximum percentage of page that can be filled with floating material	Integer

EXPLANATION:

a. The width and nomdepth characteristics specify the width and depth of the Page Area respectively.

b. The Bind Margin refers to the page edge that would be toward the binding or spine of the document if this document were bound as in a book. This edge necessarily alternates for recto and verso pages (for example, in the most common case, the right margin of verso pages and the left margin of recto pages are the bind margins). The specification of the Bind Margin determines what the quadding values of “In” and “Out” mean. If the Recto/Verso toggle is on, then for any page, “In” means toward the bind margin and “Out” means toward the margin opposite to the bind margin. If the Recto/Verso toggle is off, then for all pages (both recto and verso), “In” means toward the margin that would be the bind margin for a recto page (even if the current page is verso) and “Out” means toward the margin opposite that which would be the bind margin for a recto page. The Bind Margin characteristic can be defined only for the recto page in the Page Set. Therefore, a Bind Margin of “left” means that the left margin for recto pages—and consequently, the right margin for verso pages—will be the bind margins.

c. The Change Mark Width (chgmkwid) specifies the width of the Change Mark Area. All Change Mark Areas have the same width.

d. Change Mark Areas shall be placed according to the following rules:

    1. When Change Mark Placement (chgmkplc) is Left, the Change Mark Area appears to the left of all Column Areas by the amount of the Change Mark Offset (chgmkoff). The setting of the Recto/Verso toggle has no effect on change mark area placement in this case.
    2. When Change Mark Placement is Right, the Change Mark Area appears to the right of all Column Areas by the amount of the Change Mark Offset. The setting of the Recto/Verso toggle has no effect on change mark area placement in this case.
    3. When Change Mark Placement is In, the Change Mark Area appears to the “In” side—as defined by the “In (Bind) Margin” characteristic—of all Column Areas by the amount of the Change Mark Offset. The setting of the Recto/Verso toggle may effect the meaning of “In”, but its setting has no other effect on change mark area placement.
    4. When Change Mark Placement is Out, the Change Mark Area appears to the “Out” side—the opposite of the “In” side—of all Column Areas by the amount of the Change Mark Offset. The setting of the Recto/Verso toggle may effect the meaning of “In” (and hence, of “Out”), but its setting has no other effect on change mark area placement.
    5. When Change Mark Placement is Left/Right and the Number of Columns is 2, the Change Mark Area appears to the left of the first Column Area and to the right of the second Column Area by the amount of the Change Mark Offset. For any other value of Number of Columns, a Change Mark

Placement of Left/Right is equivalent to a Change Mark Placement of Out. The setting of the Recto/Verso toggle may effect the meaning of "In" (and hence, of "Out"), but its setting has no other effect on change mark area placement. When a border is on the page, the change mark area on the outside of the outermost column must be moved to that column's inside.

e. The Change Mark Offset characteristic always has a positive value, and is measured from the edge of the column area to the edge of the Change Mark Area in the direction specified in the previous rule.

f. The only place on the page where change marks can appear is in the Change Mark Areas.

g. The change mark placement characteristic on the flowing text area takes precedence over that on the page area.

h. The top and bottom float are used to specify those floats which are allowed to appear in the top and bottom of the flowtext area respectively. These floats will span the entire width of the flowtext area. The order of the Ids indicates positioning from the top of the page toward the bottom.

i. The Media Information (mediainfo) characteristic is a string that specifies a logical name that identifies any special processing to be performed on this page (or set of pages) by the output device. Interpretation of mediainfo logical names is beyond the scope of the output specification. If mediainfo is specified for both a specific page (for example, recto) and that page's pageset, the specific page's mediainfo will take precedence.

j. The Maximum float percent attribute specifies an approximate percentage of the page that can be taken up with material from other than the flowing text (for example, the total of all floats including footnotes). While exactly how a specific value is interpreted is left to the resolution of the composition system, the idea is that the FOSI writer can provide specifications such as "at least half the page must be composed of flowing text". The value of Maximum float percent provides merely another constraint on the composition system's placing of floats, and it may need to be ignored to satisfy other float-related constraints. In particular, it is not expected that Maximum float percent will be obeyed when floats are being forced out at the end of a scoping element (for example, at the end of a chapter) or for non-initial pages of multipage floats.

B.4.3.2.3 Page area (rectobb and versobf). The rectobb is used to define a different header, footer or page resource if one is required by a recto page with a blank back. Similarly the versobf is used to define a different header, footer or page resource if one is required by a verso page with a blank front. As an example, a recto page with a blank back may contain information in the footer in regards to the backing verso page being blank. The footer for such a recto page might read "2–5 /(2–6 Blank)".

For a recto blank back there is an implicit reference to the recto page defined in the same set and similarly for the verso blank front there is an implicit reference to the verso

page defined in the same set. A recto blank back will therefore be identical to the current recto page with the only possible exception being the content of the header and footer. The same relationship exists between a verso page and a verso page with a blank front. It is left to the output system to determine what will occur if a versobf is specified without a corresponding verso.

Note that rectobb and versobf are not page types whose explicit use can be specified for a specific page; that is, it is not possible to request that a particular page must be a rectobb or versobf page. Instead, these page type specifications are provided, much like blankpg, so that they will be used automatically by the composition system when appropriate. That is, when the composition system is about to produce (due to various composition requirements imposed on it by other composition constraints and the content of the document) a recto page that has a blank back or a verso page that has a blank front, it should use the appropriate rectobb or versobf page specification, if any, to determine the page layout for that page.

B.4.3.2.4 Page resource. The page resource (pageres) is used to manage strings and counters associated with a page. The pageres is resolved first during page model processing. See B.4.4 for information on reset, enumerate, and savetext.

B.4.3.2.5 Page specification. The page specification (pagespec) is used to specify the geometry of the subordinate layout areas.

The following characteristic applies to a given page specification.

Characteristics - Definitions:	Values - Definitions:
Page ID	ID

EXPLANATION:

a. The pgid specifies a unique ID which can be referenced via the pgidref attribute of the pageref element.

B.4.3.2.6 Top margin area. The Top Margin is defined as the area between the top edge of the Page Area and the top edge of the Header Area and runs the width of the Page Area. There is no space between the Top Margin Area and the Header Area.

Subordinate Layout Areas.
None.

Characteristics - Definitions:	Values - Definitions:
Nominal depth	Size/Distance

EXPLANATION:

a. The nomdepth characteristic specifies the depth of the top margin

B.4.3.2.7 Bottom margin area. The Bottom Margin is defined as the area between the bottom edge of the Page Area and the bottom edge of the Footer Area and runs the width of the Page Area. There is no space between the Bottom Margin Area and the Footer Area.

Subordinate Layout Areas.
None.

Characteristics - Definitions:	Values - Definitions:
Nominal depth	Size/Distance

EXPLANATION:

a. The nomdepth characteristic specifies the depth of the bottom margin.

B.4.3.2.8 Left margin area. The Left Margin is defined as the area between the left edge of the Page Area and left edge of the Flowing Text Area and runs the depth of the Page Area. There is no space between the Left Margin Area and the Flowing Text.

Subordinate Layout Areas.
None.

Characteristics - Definitions:	Values - Definitions:
Width	Size/Distance

EXPLANATION:

a. The width characteristic specifies the width of the left margin.

B.4.3.2.9 Right margin area. The Right Margin is defined as the area between the Right edge of the Page Area and Right edge of the Flowing Text area and runs the depth of the Page Area. There is no space between the Right Margin Area and the Flowing Text Area.

Subordinate Layout Areas.
None.

Characteristics - Definitions:	Values - Definitions:
Width	Size/Distance

EXPLANATION:

a. The width characteristic specifies the width of the right margin.

B.4.3.2.10 Change mark area. When Page Set Orientation is set to Portrait, the Change Mark Area is defined as an area whose boundaries are within the Left or Right Margin or Gutter Areas and runs the extent of the Flowing Text Area plus the Header and Footer Areas. When Page Set Orientation is set to Landscape, the Change Mark Area is defined as an area whose boundaries are within the Space Above, Space Below, and Gutter Areas and run the extent of the Flowing Text Area. The Header and Footer Change Mark areas remain the same for either Orientation.

EXPLANATION:

a. The Change Mark Area does not have an associated element in the OS DTD. The location of the Change Mark Area in the page model is determined by the Change Mark Area Width characteristic, the Change Mark Offset characteristic, and Change Mark placement characteristics.
b. The sum of the Width characteristic of the Change Mark Area and the Change Mark Offset characteristic should be less than the Width characteristic of all Margin areas in which the Change Mark Area may appear. When Page Set Orientation is Landscape the Depth characteristic of Space Above area or Space Below area should be greater than the characteristics of Change Mark Offset and Change Mark Width.
c. Change Mark Areas are only used to place change marks as indicated by the Change Mark Category.

B.4.3.2.11 Header and footer area (Header and Footer Area). The Header Area is defined as the area immediately below the Top Margin and above the Flowing Text Area. The Header area is bordered by the Left and Right Margin Areas. The Header Area may contain items such as publication number and security classification. Some of these items may be repetitive from page to page and some may change from page to page. Some Header information may be determined by the actual content of the page. The depth of the header can vary depending upon its contents. There is no space between the Header Area and the Top Margin Area. When the header grows past nominal, extra depth is obtained from the flowtext area with the space below remaining constant.

The Footer Area is defined as the area immediately above the Bottom margin Area and below the Flowing Text Area, and is bordered by the Left Margin area and the Right Margin Area. The Footer Area may contain items such as the folio (page number) and revision date and number. Some of these items may be repetitive from page to page, and some may change from page to page. Some Footer information may be determined by the actual content of the page. The depth of the footer can vary depending upon its contents. There is no space between the Bottom Margin Area and the Footer Area. When the footer grows past nominal, extra depth is obtained from the flowtext area with the space above remaining constant.

Subordinate Layout Areas.
None.

These characteristics apply to Header and Footer Areas.

Characteristics - Definitions:	Values - Definitions:
Nominal Depth - The preferred depth (what should occur).	Size/Distance
Maximum Depth - The greatest depth that can occur.	Size/Distance
Space Flow	Size/Distance

EXPLANATION:

a. The nomdepth specifies the depth that the composition system should strive for in all cases.
b. The maxdepth should never be exceeded.
c. The effect of specifying different values for Nominal and Maximum Depth values allows for variable spacing to be used.
d. If only Nominal Depth is specified or if it equals the Maximum value, then it is assumed the depth is fixed, that is, it is not permissible to allow Header and Footer Areas to be variable.
e. The spaflow characteristic on the Header or Footer gives a vertical amount to contribute immediately following the Header area or immediately preceding the Footer area respectively. This amount may be combined with vertical spacing contributed by the prespace or postspace of the Flowing Text area (see B.4.4.11).

B.4.3.2.11.1 Vertical quadding and text flow in headers and footers. The vertical quadding category is contained in the content model for the header and footer elements. It may occur 0 or more times in the content model of header or footer. Its occurrence separates the contents of the header of footer into groups. Within each group, text is output according to the characteristics specified for each category in the group (sectext, ruling, puttext, putgraph, usetext).

Characteristics - Definitions:	Values - Definitions:
Vertical Quadding	List (Top, Middle, Bottom)

EXPLANATION:

a. Vquad is used to specify the positioning of groups of elements placed in the header and footer area of the page.
b. Each group is positioned independently within the header and footer area according to the vquad specification for that group. The tops of all groups with a Vertical Quadding characteristic of Top are aligned to the top of the header or footer.

The middles of all groups with a Vertical Quadding characteristic of Middle are aligned to the middle of the header or footer. The bottoms of all groups with a Vertical Quadding characteristic of Bottom are aligned to the bottom of the header or footer.

c. If the first category in the header or footer is not Vertical Quadding, then the group in the header or footer from the start to the end or the first vquad is treated as though Vertical Quadding of Top had been specified for that group.

d. Nominal prespace and postspace are included in the calculation of the top, middle, and bottom of each vquad group in the header or footer.

e. The Vertical Quadding category implies a textbrk characteristic of endln="1" for the last category specified in the preceding vertical quadding group, and a textbrk characteristic of startln="1" for the first category in the following vertical quadding group.

f. It is possible to write FOSIs where the combination of specifications in a header or footer using the Vertical Quadding category will cause text to overlap. No error is detected by the output system in this case.

B.4.3.2.11.2 Security text. The security text (sectext) element is used to indicate the positioning of the security classification indication in the header or footer of a page. The security classification indication for the specified scope (page, sheet, or document) as calculated automatically by the composition system is inserted into the header or footer in place of the sectext element. See B.4.2 for a detailed discussion of the Security description.

Characteristics - Definitions:	Values - Definitions:
Scope	List (Page, Sheet, Document)
Security description reference	IDREF

EXPLANATION:

a. The scope characteristic defines the boundaries for consideration when determining the highest security level used. The security text string for the highest level contained within the scope is placed in the header or footer, as defined by the pageset in effect. If the Scope characteristic is left unspecified, “page” will be used.

b. The Security description reference (secref) characteristic indicates which of potentially multiple security descriptions should be used to determine the appropriate values. If the secref characteristic is left unspecified, the first security description (secdesc) in the same module containing this page description (or the first security description of the global module if this module contains no security descriptions) shall be used. If is an error to omit the secref characteristic if there are no secdescs in either the current module or the global module.

c. More complex security markings can be handled using the “security attribute specification” feature in the security description (see B.4.2).

d. Note it is not possible for security markings to always be automatically generated. Information which is not in itself classified, can become classified by its appearance with other information. There must always be built into any SGML publishing environment an ability to manually override FOSI generated security markings. The markover element of the secatt construct which is part of the security description section of the FOSI may be useful in implementing such an override capability. The final responsibility for security markings will always exist external to a FOSI author.

B.4.3.2.11.3 Ruling, puttext, putgraph, and usetext. See B.4.4 for information on these elements.

B.4.3.2.12 Flowing text area and its subordinate areas. The Flowing Text Area is defined as the area below the Header Area, above the Footer Area, and in between the Left Margin Area and the Right Margin Area. The flowing text area may contain items such as flowing text, figures, and tables. The Flowing Text Area may be subdivided into Column Areas that are separated by Gutter Areas. The depth of the flowing text area is equal to the difference between the depth of the page and the sum of the top margin, bottom margin, header, footer, space above, and space below. The width of the flowing text area is equal to the difference between the width of the page and the sum of the left margin and right margin.

If there is one column, text flows from the upper left to the lower right. If there is more than one column, the text will flow from the top of the left most column to the bottom of the right most column before continuing on the succeeding page and the width of the columns shall be identical.

There may, optionally, be more than one Flowing Text Area specification for a given page specification, in which case each one should be for a different Number of Columns. The various Flowing Text Area specifications would each be used when their respective Number of Columns specification matches the current number of columns being formatted (as affected by the Span characteristic of the Span category). If the value of the current number of columns is one and there is no Flowing Text Area specification for one column, an implicit one column specification that spans all columns would be assumed. If for a given number of columns there is no matching Flowing Text Area specification, it is an error and its resolution is determined by the output system.

The specification of the change mark placement (chgmkplc) characteristic on a flowtext takes precedence over the change mark placement specified in the rectopg or versopg to which the flowtext belongs.

Subordinate Layout Areas:
Column Area, Gutter Area, Footnote Area.

B.4.3.2.12.1 Flowing text area (flowtext). These characteristics apply to Flowing Text Areas.

Characteristics - Definitions:	Values - Definitions:
Number of Columns	List (1 | 2 | 3 | 4 | 5 | 6 | 7 | 8)
Top Floats - an ordered list of Float Location Id references.	IDREFS
Bottom Floats- an ordered list of Float Location Id references.	IDREFS
Balance	Toggle
Change Mark Placement	List (Left, Right, In, Out, Left/Right)

EXPLANATION:

a. Numcols specifies the number of columns.

b. The value of the Balance characteristic is ignored when the Number of Columns characteristics has a value of “1”.

c. When the Balance characteristic is turned on, it takes effect in two situations:

   1. When an element with Start Page turned on is encountered (when a new page is forced).
   2. When an element with the Keep characteristic turned on will not fit on the current page, and a new page is therefore forced.

   In these cases, the text remaining on the page is balanced within the Flowing Text Area.

d. There are no cases where some, but not all, Column Areas occurring within the same Flowing Text Area are balanced.

e. Topfloat and botfloat specify those floats which will span the entire width of the flowtext area. The order of the Ids indicates positioning from the top of the page toward the bottom. Note that due to spanning a page can have more than one flowtext area.

f. The Space Flow characteristic of the Header contributes to the amount of vertical space between the Header Area and the Flowing Text Area prior to the spacing contributed by the Prespace category contained in the flowtext’s content model. The Space Flow characteristic of the Footer Area contributes to the amount of vertical space between the Flowing Text Area and the Footer Area following the spacing contributed by the Postspace category contained in the flowtext’s content model. When a Flowing Text Area does not abut a Header or Footer Area (for example, a spanned section in the middle of a page), the Prespace and Postspace contributed by consecutive flowtext’s will be as indicated in each flowtext’s content model. The Space Flow spacing will be treated as a vertical spacing amount whose minimum, nominal, and maximum amounts are all equal to the value assigned to spaflow, whose priority is Force, and whose Conditional

is Keep. In all cases, all consecutive, uninterrupted vertical spacing contributed by Space Flow, Prespace, and Postspace will be combined according to the usual rules for resolving consecutive vertical spacing as described in the descriptions of Prespace and Postspace.

g. The chgmkplc on flowtext takes precedence over chgmkplc specified on a rectopg, versopg or blankpg.

B.4.3.2.12.2 Column area. The Column Area is used to define the columns of the flowing text area. Column characteristics are required even when there is a single column.

Subordinate Layout Areas.
None.

Characteristics - Definitions:	Values - Definitions:
Width	Size/Distance
Tolerance	Size/Distance
Vjprior	List (Column, Composition)
Top Floats - an ordered list of Float location ID references	IDREFS
Bottom Floats - an ordered list of Float location ID references	IDREFS

EXPLANATION:

a. The Width characteristic specifies the width of any single column; the width of each column is the same.

b. The Tolerance characteristic specifies the maximum amount of space that can occur between the bottom of a column and the bottom of the Flowing Text Area when columns are balanced.

c. The page immediately preceding a page caused by the Start New Page characteristic is not subject to the Tolerance requirement.

d. The vjprior (Vertical Justification Priority) characteristic specifies which set of characteristic values has precedence for purposes of vertical justification: the depth of the Column Area or the Vertical Justification Composition characteristics (Prespace, Postspace, and Keeps) specified for the elements that occur on the page.

e. The list of Float location Ids (topfloat and or botfloat) specify which float layout areas can occur at the top/bottom of the current column. The order of the Ids indicates positioning from the top of the column toward the bottom.

f. Footnotes will either come out before all Float location areas or after all of them depending on the Footnote Float characteristic of the footnote element in the Page Description.

g. It is permitted to include the same Float location ID reference in both the Top and Bottom Floats in the case of one per page floats. This indicates that the output system can place such floats into either location. Such a construct is most appropriately used with a Page Type of After Reference.

h. It is permitted to include the same Float location ID reference in both the Column specification (in the Top or Bottom Float), the Flowtext specification (in the Top or Bottom Float) and in page (recto, verso, blank) specifications in the case of once per page floats. This implies that the output system will emit floats (maintaining the first-in-first-out order) as best it can depending on the width of the floats. It should be noted that such a specification can lead to a set of constraints that cannot all be satisfied.

i. It is not permitted to include the same Float location ID reference for a Float location whose Type is Repeat at Page Break or Multi-reference more than once per page.

B.4.3.2.12.3 Prespace and postspace. See B.4.3.2.12.1 for a discussion of how these categories interact with space flow specifications in determining spacing between headers, footers, and content which spans columns. See B.4.4 for a discussion of the characteristics of these categories.

B.4.3.2.12.4 Gutter area. The Gutter Area is the space between Column Areas. Its depth is equal to the depth of the flowing text area.

Subordinate Layout Areas.
None.

Characteristics - Definitions:	Values - Definitions:
Rlthick	Size/Distance
Ruleclr	List (Black, White, Red, Orange, Yellow, Blue, Violet, Brown, Grey)
Rulepct	INTEGER

EXPLANATION:

a. When the gutter rule thickness is non-zero, a vertical rule of this thickness (in the given color) is drawn centered in each gutter.

b. Ruleclr specifies the color of the gutter rule.

c. The integer value for Rule Percent is specified as a percentage. For example, a value of “80” means 80% of black.

B.4.3.2.12.5 Footnote area. The Footnote Area is a sub-area of the Flowing Text Area. The bottom of the Footnote Area is typically immediately above the Footer Area, but its depth can vary from page to page or column to column. The Footnote Area is unique

in that although the reading direction of its contents is the same as it is for the Flowing Text Area, the placement of its contents cause the Footnote Area to grow upwards within the Flowing Text Area. If the contents of the Footnote Area does not fit within the Maximum Depth characteristic, the text flows to the next Footnote Area. Note that the characteristics specified in the subchars content of the footnote element apply only to the material generated by the various attributes of the footnote element (for example, Footnote Continue String) and not to the text of the actual footnotes themselves (see B.4.7).

Characteristics - Definitions:	Values - Definitions:
Width	List (Column, Flowing Text)
Maximum Depth	Size/Distance
Footnote Separator Thickness	Size/Distance
Footnote Separator Length	Size/Distance
Footnote Breaking	Toggle
Footnote Continue Separator Thickness	Size/Distance
Footnote Continue Separator Length	Size/Distance
Footnote Continue String	String
Space Above	Size/Distance
Footnote Float	Toggle

EXPLANATION:

a. The Width characteristic specifies whether the Footnote Area is the width of a Column or the Flowing Text Area. This determines whether the Footnote Area spans across columns when there is more than one column.

b. The Maxdepth characteristic specifies the maximum depth the Footnote Area can have. If nothing is placed in the Footnote Area, it has a depth of 0.

c. Footnote separator ruling is disabled by supplying a value of zero to the footnote separator thickness (ftnsepth) and footnote continue separator thickness (ftncntsp) characteristics.

d. Footnote separator length (ftnsepln) specifies the length of the footnote separator ruling. The footnote separator rule is drawn with a quadding of center.

e. When the Footnote Breaking characteristic is turned on, footnotes may be continued across columns or pages. If the Footnote Breaking characteristic is turned off, the footnote must be made to fit on the same Column or Flowing Text Area.

f. Footnote continue separator length (ftnconsl) specifies the length of the footnote continue separator, which is a rule whose thickness is specified by the Footnote

Continue Separator Thickness (ftnconsp). The footnote continue separator line is drawn with a quadding of center.

g. Footnote Separators shall be placed such that the top of the Separator abuts the top of the Footnote Area and extends down into the Footnote Area the amount of its Thickness.

h. The Footnote Continue String (ftnconst) value is specified using the same syntax described for the Savetext Construction Rule (see B.3.4.10). It is placed identically according to the rules for footnote separators.

i. The Space Above determines the vertical space in between the Footnote Area and the Flowing Text Area.

j. When the Footnote Float characteristic is turned on, the Footnote area appears closest to the text (in the Column or Flowing Text) preceding any Float location areas. If the Footnote Float characteristic is not enabled, the Footnote area follows all Float location areas.

B.4.3.2.13 Border specification. The Border Specification associates a name for a border used within a FOSI with an external entity that contains the actual border graphic. Borders underlay the content of the page and as such are placed on the page without regard to page content. Borders can be used to affect the appearance of any portion of the total page area.

Characteristics - Definitions:	Values - Definitions:
Border Indicator - name of the border	String
Border Entity - the identifier of the border entity	Pointer
Precedence	Integer

EXPLANATION:

a. The border specification does not indicate that any border is to appear on the pages with which it is associated; it simply identifies borders which can occur on the page.

b. For a border to appear on a page, the value specified in the bordname attribute must be specified via the bordname attribute of the border element by an e-i-c whose content is being flowed onto the page. When a Border name is referenced, the border file pointed to by the bordent attribute will appear on the page, underlaying the content of the page. As an example, an element with emergency information could trigger the appearance of an emergency border on the page its content appears on.

c. Border Entity (bordent) specifies an entity whose public or system identifier identifies a file containing an appropriate graphic to create the border on the page. The referenced graphic should be the complete border graphic appropriately sized for the page and covering the appropriate page sides. The referenced graphic

underlays any other images on the page. Note that to have a FOSI parse, there must be a NON-SGML DATA (NDATA) ENTITY declaration where the name of the entity matches the value of the bordent attribute.

d. Precedence is used to specify the order for placing borders when more than one border is to appear on the page. The border with the lowest precedence value will be placed "first" on the page (that is, will underlay all others), followed by the border with the next lowest precedence and so on. It is left to the output system to determine placement when borders have the same precedence value.

B.4.4 COMPOSITION — TEXT, STYLE DESCRIPTION CHARACTERISTICS.

B.4.4.1 Font. A collection of character images having the same basic design.

Characteristics - Definitions:	Values - Definitions:
Inherit	Toggle
Style	List (serif, sanserif, monospaced serif, monospaced sanserif)
Family Name	String
Size	Size/Distance
Posture	List (Upright, Oblique, Back slanted oblique, Italic, Back slanted italic)
Weight	List (Ultra light, Extra light, Light, Semi light, Medium, Semi bold, Bold, Extra bold, Ultra bold)
Proportionate Width	List (Ultra condensed, Extra condensed, Condensed, Semi condensed, Medium, Semi expanded, Expanded, Extra expanded, Ultra expanded)
Small Caps	Toggle
Baseline Offset	Size/Distance

EXPLANATION:

a. The Family Name value need not match one of the Style values, but should designate the actual family used, as named by the manufacturer of the font family. Family Name values need only be supplied for optional use by the receiving system. If the Family Name is non-null, then in the case of a conflict between the Family Name and the Style, the Family Name would take precedence over the Style specification by a system that uses family names.

b. Only positive values are allowed for the Size characteristic.

c. A positive value for the Baseline Offset indicates a distance above the text baseline; a negative value indicates a distance below.

B.4.4.2 Leading. The amount of space between the baselines of text lines in a single element.

Characteristics - Definitions:	Values - Definitions:
Inherit	Toggle
Leading	Size/Distance
Force Leading	Toggle

EXPLANATION:

a. Only non-negative values are allowed for Leading.
b. When the Force Leading toggle is enabled, the leading for each line will be equal to the largest leading set on an element on that line. When the Force Leading toggle is not enabled, the leading will be equal to the first leading set in an element in which no children specify a new line (via Textbrk's startln or endln); however, the output system will increase the leading as required on any given line to avoid overprinting of text.
c. Leading applies to all lines of text including the first line of an element.

B.4.4.3 Hyphenation. The process of breaking a word at the end of a line of text for purposes of line justification.

Characteristics - Definitions:	Values - Definitions:
Inherit	Toggle
Language	IDREF
Hyphenation - Disallowed (0) or allowed (1).	Toggle
Hyphenation Zone - Ragged justification amount	Size/Distance

EXPLANATION:

a. The Hyphenation Zone specifies the maximum amount of space from the margin that can be left when doing ragged justification (aesthetic rag). This characteristic only has an effect for the Quadding category's Quad characteristic values of right, left, in, and out. In these cases, the zone amount gives the maximum distance from the ragged margin that the text should be set. The Hyphenation Zone is ignored for the last line of a paragraph. The effect of this characteristic is independent of the setting of the Hyphenation characteristic.

B.4.4.4 Wordspace. The white space between words that may be adjusted for readability and line justification.

Characteristics - Definitions:	Values - Definitions:
Inherit	Toggle
Minimum - The least amount of space that can appear.	Size/Distance
Nominal - The preferred amount of space (what should appear).	Size/Distance
Maximum - The greatest amount of space that can appear.	Size/Distance

EXPLANATION:

a. The effect of specifying different values for Minimum, Nominal, and Maximum allows for variable spacing to be used for horizontal justification.
b. When the values for Minimum, Nominal, and Maximum are equal, the spacing is fixed (not variable).
c. Only positive values shall be specified, typically with em spaces.
d. If no values are specified, inherited, or defaulted at any level, the composition system will determine defaults in its own manner, potentially related to the current font metrics.

B.4.4.5 Sentence space. In some styles, the adjustable white space between sentences (that is, following sentence-ending punctuation) may be greater than the usual word space. The extra end-of-sentence space (sentxsp) specifies the extra amount of adjustable space that is added between sentences in addition to the word space in force. It is left to the output system to determine when to apply this extra spacing (that is, what spaces are “between sentences”).

Characteristics - Definitions:	Values - Definitions:
Inherit	Toggle
Minimum - The least amount of extra space that can appear.	Size/Distance
Nominal - The preferred amount of extra space (what should appear).	Size/Distance
Maximum - The greatest amount of extra space that can appear.	Size/Distance

EXPLANATION:

a. The effect of specifying different values for Minimum, Nominal, and Maximum allows for variable spacing to be used for horizontal justification. Note that all these values are used in addition to those for the current word spacing in effect, to determine the amount of space allowed between sentences.
b. When the values for Minimum, Nominal, and Maximum are equal, the spacing is fixed (not variable). When all these values are zero, spacing between sentences is the same as that between words.
c. Only non-negative values shall be specified, typically in terms of em spaces.
d. If no values are specified, inherited, or defaulted at any level, the composition system will determine defaults in its own manner, potentially related to the current font metrics.

B.4.4.6 Letterspace. White space between letters in a word which may be adjusted for readability and line justification.

Characteristics - Definitions:	Values - Definitions:
Inherit	Toggle
Minimum - The least amount of space that can appear.	Size/Distance
Nominal - The preferred amount of space (what should appear).	Size/Distance
Maximum - The greatest amount of space that can appear.	Size/Distance
Kerntype - allowed types of kerning.	List (None, Pair, Track, Sector, Pair/Track, Track/Sector)
Kerning Pairs - specification of table of kerning pairs.	Pointer

EXPLANATION:

a. The effect of specifying different values for Minimum, Nominal, and Maximum allows for variable spacing to be used for horizontal justification.
b. When the values for Minimum, Nominal, and Maximum are equal, the spacing is fixed (not variable).
c. The Size/Distance values shall have positive values, typically em spaces.
d. Kerntype specifies the method of kerning to be used by the composition system. Pair kerning specifies an approach whereby kerning pairs are looked up in a kerning table. Track kerning is a methodology for placing the same amount of space between characters in a line. Sector kerning is an algorithmic approach

for placing space between characters depending on the characters in the line. Combinations of pair and track kerning and track and sector kerning may be specified.

e. If no values are specified, inherited, or defaulted at any level, the composition system will determine defaults in its own manner, potentially related to the current font metrics.

B.4.4.7 Indent. Positioning along the writing direction.

Characteristics - Definitions:	Values - Definitions:
Inherit	Toggle
Left Indent - Placement of a text margin relative to the left boundary of the current Layout Area.	Size/Distance
Right Indent - Placement of a text margin relative to the right boundary of the current Layout Area.	Size/Distance
First Line - Special left Indent value for first line of content.	Size/Distance

EXPLANATION:

a. See B.3.4.9 for the discussion of Indent syntax.

b. For Left and First Line Indents, a positive value positions the text margin the specified distance to the right, and is the default. A negative value specifies an indent to the left.

c. For a Right Indent, a positive value positions the text margin to the left and is the default. A negative value specifies an indent to the right.

d. Indent values may specify a position outside of the current flowing text area, for example, in the margins. This is permitted, though the results may be output system dependent.

e. Indent specifications are generally ignored (and current indentations are maintained) for “inline” e-i-cs (see B.3.9). Furthermore, the First Line indent is used only for the first line of an element even if that element is “interrupted” by a descendent element with startln="1" or endln="1" and the remaining character content of the interrupted element uses that element’s original left indent.

B.4.4.8 Horizontal quadding. The horizontal alignment of text lines within an element with respect to the Layout Area boundaries.

Characteristics - Definitions:	Values - Definitions:
Inherit	Toggle
Horizontal Quadding	List (Right - Flush right/ragged left. Left - Flush left/ ragged right. Center - Centered. In - Flush to inside margin/outside ragged. Out - Flush to outside margin/inside ragged. Justify - Flush left & flush right. As is - Place lines exactly as in the source.)
Lastline - How to align the last text line of the element.	List (Right - Flush right/ragged left. Left -Flush left/ragged right. Center - Centered. In - Flush to inside margin/outside ragged. Out - Flush to outside margin/inside ragged. Justify - Flush left & flush right. Relative - see note c.)

EXPLANATION:

a. In and Out quadding values refer to the bind edge and the edge opposite the bind edge, respectively (see B.4.3.2.2).
b. The “asis” value specifies that space characters and line breaks within the text be preserved and no additional space characters or line breaks be added (see B.3.14).
c. When a value for Lastline is “relative”, the Lastline is aligned according to the value of Quadding, except when Quadding is “justify” in which case Lastline is aligned “left”.
d. Quadding specifications are generally ignored unless a new line is triggered by a startln="1" or by an endln="1". In mixed content models where the child’s content is not separated from the parent’s content by a startln or endln, the parent’s quadding specifications are generally used. If an element’s character content is interrupted by an element with a startln="1" or endln="1" or a puttext, usetext, or putgraph that start or end a line, the last line of #PCDATA prior to the intervening output uses the lastquad of the quadding specification (see B.3.9).

B.4.4.9 Highlight. Special presentation characteristics for text.

Characteristics - Definitions:	Values - Definitions:
Inherit	Toggle
Foreground/Background - Reverse color	Toggle
Scoring - Overlay rules on the text line.	Integer
Score Weight - Weight of the rule.	Size
Score Offset - Offset of the base of the rule to the text baseline.	Size

Score Character - Character to be overlaid on the text.	String
Use Score Character - Whether to overlay using the score character	Toggle
Background Color - Color of the area between text baselines.	List (Black, White, Red, Orange, Yellow, Green, Blue, Violet, Brown, Gray)
Foreground Color - Color of the text characters	List (Black, White, Red, Orange, Yellow, Green, Blue, Violet, Brown, Gray)
Background Screen - Shading of the area between text baselines.	Integer
Foreground Screen - Shading of the text characters	Integer
All Caps	Toggle
Score Spaces - Whether spaces should be scored/overlaid	Toggle

EXPLANATION:

a. The Scoring, Score Weight, and Score Offset characteristics are used together to create rules overlaid with the text, for example underlines and overlines. To achieve score offset, a positive value will place the score below the baseline, and a negative value will place the score above the baseline. The Score Character may additionally be specified to overlay a different character on the text, for example an “x” for a “strike out” effect.

b. If the Use Score Character (scorechron) characteristic is “on”, the Score Character (if any is specified) is used to overlay the text. If the Use Score Character (scorechron) characteristic is “off”, no overlaying by the Score Character is done despite the setting of the Score Character characteristic.

c. The integer value for Foreground and Background Screen is specified as a percentage. For example a value of “80” means 80% of black. Note that a Foreground Screen value of “0” means that the text is not visible.

d. The value of scoring indicates the multiplicity of the score rule. A value of zero means there is no scoring (and the Score Weight and Score Offset are ignored). A value of one means to score with a single rule (whose weight and offset are determined by Score Weight and Score Offset). A value of two means to score with a double rule (where each individual rule’s weight is given by Score Weight, Score Offset measures the offset to the base of the lower of the two rules, and the gap between the two rules is left to be resolved by the output system).

e. If the Score Spaces toggle is “off”, spaces (blanks) in the highlighted text are not overlaid with a rule or score character. If the Score Spaces toggle is “on”, spaces should be scored.

B.4.4.10 Change mark. The data that appears in the Change Mark Area, triggered by the occurrence of a particular element.

Characteristics - Definitions:	Values - Definitions:
Literal - A literal that should appear instead of a bar.	String
Bar Thickness - The thickness of the Change Bar.	Size/Distance
Join - Join bars of contiguous lines of marked text.	Toggle
Type - Whether this element starts ends, or contains the region to be marked.	List (Content, Start, End)
Class - an identifier to pair matching Start and End type change mark specifications	Name token
Offset - offset of bar within change mark area	Size/Distance

EXPLANATION:

a. Either a bar thickness of greater than zero or a literal should be specified. If both are specified, the bar should appear and not the literal.
b. The bar is positioned in the Change Mark Area with its bottom aligned to the baseline of the text to which it corresponds. The height of the bar equals the leading specified for the element. The Bar Thickness indicates the width of the bar.
c. When a bar is used for the change mark, if Offset is not specified, the bar is centered horizontally in the change mark area. If Offset is specified, it is used as the distance from the edge of the change mark area nearest the associated flowing text and the (center of the) change bar. The Offset characteristic is ignored when the change mark is not a bar.
d. A literal is positioned in the Change Mark Area with its baseline aligned to the baseline of the text to which it corresponds. The total width of the literal should be less than that of the Change Mark Area (that is, the literal is expected to fit on one line).
e. When Join is turned on, bars of adjacent lines of text, even though they may be separated by some space in addition to the leading, should be extended so that they join to form one contiguous bar.
f. The Type characteristic indicates whether this e-i-c starts, ends, or contains the region to be marked with the specified change mark. If Type is Content, then

the change mark is in effect for the duration of the current element. If Type is Start, then the change mark goes into effect at the beginning of the current element and stays into effect until the end of an element whose Change Mark Type is End. (Change Mark Types of Start and End allow the use of individual empty elements for the delineation of change regions.)

g. The Class characteristic provides an identifier that allows the classification of Start/End change mark specifications so that occurrences of e-i-c instances using the Start/End change mark type can be properly paired even when they are nested. An e-i-c instance with Type="End" will only close one with Type="Start" if the values of their Class characteristics are identical. Note that it remains a requirement that all uses of elements associated with Start/End change mark specifications in the document instance are properly paired and nested with respect to each other. The Class characteristic is ignored when Type="Content".

h. When a change mark is specified for an e-i-c and the Type characteristic is Content, the change mark continues for the duration of the element including all its children. When a change mark is specified for an e-i-c and the Type characteristic is Start, the change mark continues until an e-i-c with a Change Mark Type equal to End is encountered; the change mark continues for all parts of all elements and their children between the start and end point. The change mark is placed in the change mark area adjacent to the flowing text area that contains the content marked as changed. The change mark does not appear in the change mark area adjacent to headers and footers—even if the changed region continues from one page onto another—unless there is a use of the change mark category in the header or footer itself.

i. When any children of a region that has a Change Mark float, they will continue to be marked by the change mark.

B.4.4.11 Prespace. The amount of space in the vertical direction added before an element (in addition to its leading).

Characteristics - Definitions:	Values - Definitions:
Minimum - The least amount of space that can appear.	Size/Distance
Nominal - The preferred amount of space (what should appear).	Size/Distance
Maximum - The greatest amount of space that can appear.	Size/Distance

Conditional - Indicates whether to keep or discard space at a column or page break.	List (Keep, Discard)
Adjacency Priority - Indicates a preference for choosing one element's spacing requirements over another's.	List (Force, High, Medium, Low, None)

EXPLANATION:

a. The effect of specifying different values for Minimum, Nominal, and Maximum allows for variable spacing to be used for vertical justification. Spacing values for a given prespace or postspace specification may be negative; such specifications are combined with other adjacent specifications as described in note c. A final effective negative vertical spacing implies that the output system should move upward on the current page.

b. When the values for Minimum, Nominal, and Maximum are equal, the spacing is fixed (not variable).

c. Prespace and Postspace values are not normally additive. Whether to use an element's Prespace or the preceding element's Postspace as the spacing between two elements is determined by the Adjacency Priority value. The value from the element with the highest Adjacency Priority is used. When the Adjacency Priority values are equal, the element with the larger Nominal value is used. If the Adjacency Priority values are equal and the Nominal values are equal, the given Nominal value, the maximum of the Minimum values, and the minimum of the Maximum values will be used for the effective spacing's Nominal, Minimum, and Maximum values respectively. When the Adjacency Priority values both equal "Force", the effective spacing's Nominal, Minimum, and Maximum values will each be the sum of the respective Nominal, Minimum, and Maximum value pairs.

d. Either through defaulting or by explicit specification, an e-i-c may have either a significant prespace and startln="0" or a significant postspace and endln="0". For example, an e-i-c instance with startln="0" might be expected to contribute its prespace depending on whether a subsequent e-i-c instance has startln="1". Therefore, pre/postspaces (and their adjacency priorities) of all e-i-c instances that occur since the last typeset material (whether from content text or generated material) up to the next typeset material must be compared to determine what value would be used; then, vertical spacing of this amount is actually used if and only if any of the intervening e-i-c instances had specified a new line via startln="1" or endln="1" (either explicitly or via defaulting).

e. The priority characteristic is used to determine which element's characteristics and values take precedence when there is a conflict. The priority characteristic range of values in descending order is: "Force", "High", "Medium", "Low", and "None". Normally a simple comparison of these values will provide a

solution to the conflict. Where priority characteristic values are equal, there are additional precedence rules that are applied depending on where the characteristic is encountered.

B.4.4.12 Postspace. The amount of space in the vertical direction added at the completion of an element (in addition to its leading).

Characteristics - Definitions:	Values - Definitions:
Minimum - The least amount of space that can appear.	Size/Distance
Nominal - The preferred amount of space (what should appear).	Size/Distance
Maximum - The greatest amount of space that can appear.	Size/Distance
Conditional - Indicates whether to keep or discard space at a column or page break.	List (Keep, Discard)
Adjacency Priority - Indicates a preference for choosing one element's spacing requirements over another's.	List (Force, High, Medium, Low, None)

EXPLANATION:

a. The effect of specifying different values for Minimum, Nominal, and Maximum allows for variable spacing to be used for vertical justification. Spacing values for a given prespace or postspace specification may be negative; such specifications are combined with other adjacent specifications as described in note c. A final effective negative vertical spacing implies that the output system should move upward on the current page.

b. When the values for Minimum, Nominal, and Maximum are equal, the spacing is fixed (not variable).

c. Prespace and Postspace values are not normally additive. Whether to use an element's Prespace or the preceding element's Postspace as the spacing between two elements is determined by the Adjacency Priority value. The value from the element with the highest Adjacency Priority is used. When the Adjacency Priority values are equal, the element with the larger Nominal value is used. If the Adjacency Priority values are equal and the Nominal values are equal, the given Nominal value, the maximum of the Minimum values, and the minimum of the Maximum values will be used for the effective spacing's Nominal, Minimum, and Maximum values respectively. When the Adjacency Priority values both equal "Force", the effective spacing's Nominal, Minimum, and Maximum values will each be the sum of the respective Nominal, Minimum, and Maximum value pairs.

d. Either through defaulting or by explicit specification, an e-i-c may have either a significant prespace and startln="0" or a significant postspace and endln="0". For example, an e-i-c instance with startln="0" might be expected to contribute its prespace depending on whether a subsequent e-i-c instance has startln="1". Therefore, pre/postspaces (and their adjacency priorities) of all e-i-c instances that occur since the last typeset material (whether from content text or generated material) up to the next typeset material must be compared to determine what value would be used. Vertical spacing of this amount is actually used if and only if any of the intervening e-i-c instances had specified a new line via startln="1" or endln="1" (either explicitly or via defaulting).

e. When an unconditional page break is specified (using Textbrk's Start Page characteristic), the page preceding the page break is usually filled at the bottom with white space. However, when the Conditional characteristic of the Postspace that is actually used (according to note c) just previous to the page break is "Keep", the requested Postspace should be used at the bottom of the page preceding the break without any additional fill. For example, if immediately preceding a page break one specifies a Postspace with an Adjacency Priority of "Force" a Nominal of 0, and a Conditional of "Keep", the last piece of text on the page preceding the break will be placed at the bottom of the page, and the rest of the page contents will be "spread out" evenly using the vertical justification flexibility provided by the various composition parameters (that is, all the Pre- and Postspace on the page will tend toward its Maximum values so as to fill the page).

f. The priority characteristic is used to determine which element's characteristics and values take precedence when there is a conflict. The priority characteristic range of values in descending order is: "Force", "High", "Medium", "Low", and "None". Normally a simple comparison of these values will provide a solution to the conflict. Where priority characteristic values are equal, there are additional precedence rules that are applied depending on where the characteristic is encountered.

B.4.4.13 Keeps. Conditions for controlling or disallowing the breaking of elements over column or page boundaries.

Characteristics - Definitions:	Values - Definitions:
Keep - Keep the whole element content together within the column, page, or line boundary.	Toggle
Scope - The scope in which to keep the element together.	List (Column, Page, Line)

Widow Count - The number of lines to keep at the top of a column.	Integer
Orphan Count - The number of lines to keep at the bottom of a column.	Integer
Keep Next - Keep with next element.	Toggle
Keep Previous - Keep with previous element	Toggle
Keep Floats Out - Specifies what category of floats should not be permitted to interrupt the pagination of the current element.	IDREFS

EXPLANATION:

a. Keep conditions can be nested (treated like a stack). A Column Keep = Off condition of a child does not override a Column Keep = On of a parent. That is, On always takes priority. The ending of a nested Keep does not turn off a parent's Keep. (Parent and child Widow Count and Orphan Count interact as any other characteristics rather than in this special pre-emptive manner.)

b. The Scope indicates the boundaries across which the element cannot break, either the column boundary, page boundary, or line boundary. Specifying "column" implies also keeping within the page. A Scope value of Line implies the output system should inhibit line breaks within the element if feasible; how feasibility is determined and what happens when infeasible is left to be determined by the output system.

c. In the event that element content must break across a column boundary, the Widow Count and Orphan Count specify the minimum number of lines that must remain at the top and bottom of the column, respectively.

d. The "next" element is the element (or part of an element) that next contributes content to the same Layout Area (Flowing Text Area or Footnote Area) in the output stream (whether this content is due to character content directly from the source or via the action of a Usetext, Puttext, Putgraph, or Character Fill). Note that one or more ancestors of the current element may end and one or more elements may begin (and maybe end) before more content is contributed to the output stream. An indication of Keep Next on the current element would imply that a break cannot occur between the last contributed content of the current element and the next contributed content in the same Layout Area of the output stream. Note that when an element's character content is interrupted by other elements (mixed content models), Keep Next applies to the last character content of the element's character content.

e. The "previous" element is the element (or part of an element) that last contributed content to the same Layout Area (Flowing Text Area or Footnote Area) in the

output stream (whether this content was due to character content directly from the source or via the action of a Usetext, Puttext, Putgraph, or Character Fill). Note that one or more ancestors of the previous element may end and one or more elements may begin (and maybe end) before the start of the current element without more content getting contributed to the same Layout Area of the output stream. An indication of Keep Previous on the current element would imply that a break cannot occur between the last contributed content in the output stream and the first contributed content put into the same Layout Area of the output stream by the current element.

f. Keep Next/Previous indicates that a break cannot occur immediately following/preceding the element's content (between this element and the next/previous element). For Scope of Column, this means that neither a column or page break may occur between the current element and the next/previous one. For Scope of Page, a page break may not occur between the current element and the next/previous one. For Scope of Line, a line break may not occur between the current element and the next/previous one. Note that, unless the Keep characteristic itself is enabled, enabling Next or Previous, while preventing a break between the current element and the next/previous one, does not prevent a break within the current element itself.

g. See the Vertical Justification characteristic for information on how Keep Characteristics interact with other characteristics that affect vertical justification.

h. The Keep Floats Out characteristic's value is a list of ID references to Float Location Ids. All pending floats associated with a Float Location whose ID is included in the Keep Floats Out characteristic will remain pending (that is, will not be permitted to be placed into the output stream) for the duration of the current element and all its children. This allows, for example, for one to disallow floating graphics from interrupting the pagination of a multipage table. Whether the floating graphic gets placed into the output stream before or after the table—and exactly where it gets placed—depends on the complete set of floating constraints. Note that floats with non-flexible page types such as same, next, facing, and frontback can easily provide constraints that, in combination with the Keep Floats Out characteristic use, are impossible to satisfy. Note also that there is no implicit interaction with the Span category and the Keep Floats Out characteristic; unless the Keep Floats Out characteristic is used to inhibit certain floats for the duration of a spanned element, it is possible for floats to interrupt the pagination of a spanned element. In this case, when the floated element is column wide and the span creates a different number of columns, it is possible that the set of floating constraints will not be satisfiable or that the result may not be what the user expects.

i. When a sequence of elements without a startln="1" or endln="1" occurs and scope="col" or scope="page", the keep specifications for the first element are used for all subsequent elements in the sequence.

j. When an element's character content is interrupted by other elements with a startln="1" or endln="1" (mixed content models), the widow and orphan conditions are used for each character content segment.

B.4.4.14 Vertical justification. The priority information for each characteristic necessary for making decisions for purposes of vertical justification.

Characteristics - Definitions:	Values - Definitions:
Inherit	Toggle
Prespace Priority	List (Force, High, Medium, Low, None)
Postspace Priority	List (Force, High, Medium, Low, None)
Keep Priority	List (Force, High, Medium, Low, None)

EXPLANATION:

a. Where it is not possible to honor all elements' Prespace, Postspace, and Keep characteristic values for the purposes of vertical justification on a column or page, a comparison of the priority characteristics listed determines which characteristics shall prevail. Where Priority values are equal, the order of occurrence shall be used to determine priority, with the preceding elements taking precedence.

B.4.4.15 Textbreak. Characteristics for allowing controlled interruptions of normal text flow.

Characteristics - Definitions:	Values - Definitions:
Start Column - Start at beginning of new column.	Toggle
Start Page - Start at the beginning of a new page.	List (Off, Verso, Recto, Next)
Resume Page - The type of page with which to resume the previous page model (interrupted by a local page model).	List (Verso, Recto, Next)
Page Model ID - The Page Model to use when starting the page	IDREF
New Page Model - Use the new page model.	List (None, Global, Local)

Start Line - Start at beginning of a new line.	Toggle
End Line - At the end of the current element, prepare to start on a new line.	Toggle

EXPLANATION:

a. Start Column, Start Page, End Line, and Start Line do not cause vertical spacing, but imply a logical start beginning a new column, page, or line respectively. That is, they work in conjunction with other text positioning characteristics, such as leading and prespace, to indicate that the next text to be placed begins at the new column, page, or line, but do not indicate any additional leading or spacing.
    1. If the current element has endln="1" and the next element has startln="1", then prespace and postspace are used as determined by the values described in the description of prespace and postspace and the leading of the next element is used between the last line of the current element and the first line of the next element.
    2. If the current element has endln="1" and the next element has startln="0", then prespace and postspace are used as determined by the values described in the description of prespace and postspace and the leading of the next element is used between the last line of the current element and the first line of the next element.
    3. If the current element has endln="0" and the next element has startln="0", then no prespace or postspace is used. If the Force Leading toggle is enabled, the leading for each line will be equal to the largest leading set on an element on that line. If the Force Leading is not enabled, the leading will be equal to the first leading set until a new line is specified (via startln="1" or endln="1"), and the output system will use whatever leading is required to avoid overprinting of text.
    4. If the current element has endln="0" and the next element has startln="1", then prespace and postspace are used as determined by the values described in the description of prespace and postspace and the leading of the next element is used between the last line of the current element and the first line of the next element.

b. When Start Column is enabled, the “next” column is the column to the right of the current column, or, if the current column is the rightmost column, the “next” column is the leftmost column on the next page.

c. Start page has the following options:
    1. When start page is equal to recto, the recto page model of the appropriate triple will be used. If this specification causes the previous verso to be blank, then the last formatted recto page will be reformatted on the recto with a

blank verso page model of the current triple. If this specification causes a recto page with blank back to be produced and this page set "triple" contains a rectobb specification, then this rectobb specification will be used for this recto page with blank back.

2. When start page is equal to verso, the verso page model of the appropriate triple will be used. If this specification causes the previous recto to be blank, then the last formatted verso page will be reformatted on the verso with a blank recto page model of the current triple. If this specification causes a verso page with blank front to be produced and this page set "triple" contains a versobf specification, then this versobf specification will be used for this verso page with blank front.

d. When the Start Page characteristic is enabled, the Start Column and Start Line characteristic values are automatically enabled. When the Start Column characteristic is enabled, the Start Line characteristic value is automatically enabled.

e. A value specified for the Page Model ID is meaningful only when the New Page Model characteristic is set to global or local and the Start Page characteristic is enabled. The New Page Model value of Global means that the page model specified by the Page Model ID will become the current page model, and will remain in effect until another Textbrk category is used. The new Page Model value of Local means that the page model specified by the Page Model ID will become the current page model until the processing of the current element is completed. A new page is started when processing of the element is completed, and the Page Model previously in effect becomes the current Page Model.

f. When a local page model ends, the type of new page that is started is determined by the Resume Page characteristic.

g. Setting End Line on an element causes a new line to be started before the first text that follows the current element's end is placed, whether what immediately follows the current element's end is text or another element.

B.4.4.16 Span. Positioning of an element across all columns on a page.

Characteristics - Definitions:	Values - Definitions:
Span - Specifies the number of columns into which to format the flowing text.	Integer
Page flow - gives a preference for how to balance material when spanning	List (L, Z)

EXPLANATION:

a. If the Span characteristic is zero, the Span category has no effect; whatever spanning (or lack thereof) currently in effect continues. If the Span characteristic

is a positive integer, it specifies the number of columns into which to format the Flowtext (using the Flowing Text Area specification in the current page specification whose Number of Columns specification matches the number of columns specified by this Span characteristic). That is, all text currently on the page is first positioned on the page as indicated by the Page Flow characteristic; then, the entire Flowing Text Area is “spanned” and then redivided into the number of columns specified by the Span characteristic. Therefore, if span="1", the behavior is to span the entire page with one column. The effect of the span is scoped by the element that spans; when that element ends, the effect of returning to the parent element is the same as starting a new spanned area with a Span characteristic value equal to that of the parent.

b. If the Page Flow characteristic has the value “Z”, when a change of span is encountered while doing multicolumn flow (that is, when span is greater than one), the flowing text encountered on the page thus far is balanced over all of the current number of columns before the span takes effect (the flow of text creates a pattern across the page leading to the term “Z-paging”). If the Page Flow characteristic has the value “L”, the flowing text encountered on the page thus far is not balanced but is all placed in the first column (and subsequent columns if necessary) before the span takes effect (“L-paging”). If the material preceding the span fits in the first column such that there is room below this material on this page for the span to take effect, the spanned material appears on this page leaving all but the first column in the flowing text preceding this span empty. If the material preceding the span does not all fit in the first column, then it fills the first column and continues in subsequent columns until it is exhausted, remaining columns are left empty (and unbalanced), and the span starts on the top of the flowing text on the next page. If the Page Flow characteristic is not specified (either explicitly or by defaulting), the default behavior shall be as if “Z” were specified.

c. There are no cases where some, but not all, Column Areas may be spanned (though the spanned area may be redivided into more than one column).

d. If a non-zero value of Span is specified for an element, the Textbrk’s Start Line characteristic is treated as if it were set to “on”.

B.4.4.17 Border. The pattern that appears along page boundaries, triggered by the occurrence of a particular element.

Characteristics - Definitions:	Values - Definitions:
Border Name - Name of the border	String

EXPLANATION:

a. The Border Name must be a Border Indicator defined in the Border Specification for the page on which the element falls. The graphic referenced by the Border Entity currently associated with the Border Indicator appears on the page.
b. The border graphic will underlay the page.
c. When a border is specified for an e-i-c, the border begins on the page where that element starts. The logical border repeats on each page during the duration of that element (including children) up to and including the page where the element is ended. (Note that a given logical border as specified by a border name may be associated with different actual border graphics for different pages.)
d. When more than one border is referenced on a page, the bordspec's precedence attribute is used to determine precedence. If the precedence attribute is not used or does not disambiguate the situation, it is up to the output system to determine which border(s) appear and in what order.

B.4.4.18 Float. This category indicates that an element will not be paginated in input document order but will instead float to some other location in the output instance where that location will be determined by the output system according to a specified set of constraints.

Characteristics - Definitions:	Values - Definitions:
Float location ID reference	IDREF
Width - the width to which to compose the floated material	List (Page, Column)
Widow height - minimum height of part of breakable float in the last floatloc area in which this float appears	Size/Distance
Orphan height - minimum height of part of breakable float in the first floatloc area in which this float appears	Size/Distance
Scope - the ancestral element(s) that scope the current float	Name(s) of element(s)
Page Type - the page(s), relative to the current page, to which this material should float	List (Same, Facing, Front/Back, Next, Forward, After Reference, Inline)
Inline - gives a preference for handling material that may either float or be inline	List (Never, Unbroken, Unsplit, Unframed)
Multi-reference Name - an identifier to distinguish this multi-reference float	NMTOKEN

EXPLANATION:

a. Specifying float characteristics in a charlist indicates that the contents of the associated e-i-c, as well as any material generated by ruling, puttext, putgraph,

or usetext, appears in the float layout area specified by the Float location ID reference.

b. The Width characteristic value (page or column wide) indicates the width to which the material to be floated is composed. The value that is used is the page or column width that is current at the time the floating e-i-c instance is encountered in the input. It is possible to specify a composition width for a floating element that is wider than the float location area into which it is destined—as with all other float constraint problems, the output system may issue an error or may employ some other fallback resolution algorithm.

c. The contents of a floating element may be breakable (as determined by the Keeps category and other composition constraints). However, even for breakable floats, it may be desirable to specify a minimum amount of the beginning or end of the floating element that must appear before or after such a break. The Widow height and Orphan height characteristics allow the specification of the minimum height of the part of the breakable float that must occur in the first (orphan) or last (widow) floatloc area in which any part of this float appears. If either of these two characteristics are left unspecified or are set to a size/distance of zero, there will be no restriction on the size of any parts of the breakable float.

d. The Scope characteristic specifies the element(s) that delimit the range for this float. Scope uses the same mechanism as “attloc”; that is, each element named in the Scope characteristic refers to an ancestor of the position of the current e-i-c in the input document. When the element named by Scope terminates (that is, when its end tag is encountered), the Scope ends. Note that Scope gives a list of element names—the first ancestral end tag encountered terminates the scope. If unspecified for a given float, the scope is the entire document.

e. When the end of any scoping element is reached, all floats scoped by that element are “flushed” (in the same sense as when a Float location ID is specified in a Reset Control list). For Float Type of Once and Multi-Reference, flushing means the floats are forced out into the output stream following any material the current e-i-c would supply to the output stream (but preceding any savetext, usetext, puttext, ruling, putgraph with placement equals after) even if this would violate the floating constraints. (This allows, for example, for containing floating within chapters.) For Float Type of Repeat at Page Break, the float locations are emptied without contributing anything further to the output stream.

f. Floating an element into a Float location whose Float Type is set to Repeat at Page Break causes the contents of the e-i-c to occur in the specified float layout area each time the float’s scoping element breaks across a page. (The specified float layout area must be allowed on the page.) On the page the scoped element starts, only bottom float layout areas are filled. On the page the scoped element ends, only top layout areas are filled. On pages spanned by the scoped element, both top and bottom float layout areas are filled.

g. In float, the Page Type characteristic specifies on which page the specified float layout area can occur for floats associated with floatlocs with float type of “Once”.

Values for this characteristic are relative to the location where the e-i-c occurred in the source document (its reference). Possible specifications are Same (same page as the reference), Facing (page facing the page on which the reference occurred), Front/Back (opposite side of same sheet as the reference), Next (the page following that on which the reference occurred), Forward (the next available page including anywhere on the same page as the reference), After Reference (the next available page including on the same page as the reference, provided the float would end up after its reference), Inline (see explanation note h). The Page Type specifies the type of first (or only in the case of multipage floats) page on which the float will appear. Page type is ignored for floats associated with floatlocs with float types of “Repeat at page break”, “Repeat at column break”, or “Multi-reference”.

h. A float Page Type of Inline means that the element should be placed, if possible, into the output stream at the point of its reference—that is, it should not float. However, if it is not possible to place the material inline given all the operative constraints, then the element should be floated as if its Page Type were specified as After Reference (and the rest of the characteristics in the float category are as specified). The two major constraints that determine whether the material is placed inline or floated are:

    1. Given that the element may be unbreakable (as indicated by the Keeps category) or may at least have an initial region that is not breakable over a page break, whether it can fit inline on the page without violating the page layout flexibility allowed by the Keeps, Prespace, Postspace, and Vertical Justification Information categories.
    2. Given that all floats associated with the same Float location as this element must be placed into the output stream in first-in-first-out order, whether placing this element inline will violate this order (in practice, this generally means that if an element associated with the same float location is currently pending, then this element may not be placed inline).

i. When a charlist contains the float category, this indicates that a possible result for the given e-i-c’s contents is to float. Whether the element actually floats is constrained by the float location, Page Type, and other factors; the keeps category in the charlist influences whether the (possibly floating) material can be broken over multiple pages (if typeset inline) or split over multiple float locations (if it floats). It may not be possible to float the material while obeying all constraints (for example, when the material itself takes up more than one page and the Page Type is same or next) or floating may not be the most desirable outcome.

    The Inline characteristic allows the specification of preferences on how to handle situations that include options of either floating the material or placing it inline. As in all cases of automatic placement of floats, it still remains possible to over-constrain the situation, and final float resolution is left to the output system.

    If the Page Type characteristic is “inline”, the Inline characteristic is ignored (and the behavior is as if inline="unbroken"). For any other value of Page Type,

the Inline characteristic is considered (as well as the Page Type). The Inline characteristic has four possible values; note that “unbroken” refers to avoiding breaking inline material over consecutive pages, “unsplit” refers to avoiding splitting a float over multiple float locations, and “unframed” refers to avoiding having text both precede and follow the (inline) material on the same page.

1. Inline="never" implies that floating is always desired; it is preferable that the float not be split (though it may be split if no other resolution is allowed by the various constraints). The Page Type characteristic is obeyed. This is the default behavior for the Inline characteristic when the Float category is specified and the Page Type is not “inline”.
2. Inline="unbroken" indicates that it is preferable for the material to be placed inline if it fits on the current page in its entirety. If not, then if the material can remain unsplit while floating (and obeying Page Type), then that is done. However, if neither are possible, the material is placed inline (and breaks over subsequent pages as necessary). This is the behavior for the Inline characteristic when the Float category is specified and the Page Type is “inline”. Note that this case differs from inline="unsplit" in that inline is considered first, and floating is the first level fallback (while the final fallback is inline over multiple pages).
3. Inline="unsplit" specifies that the material should float if it can do this both obeying Page Type and without splitting. If this is not possible, then the material is placed inline; breaking the inline material over subsequent pages is discouraged but permitted. Note that this case differs from inline="unbroken" in that floating is considered first, and inline is the fallback.
4. Inline="unframed" indicates that the material should float if it can do this both obeying Page Type and without splitting. If this is not possible, then the material is placed inline; furthermore, if in so doing, the material fits unbroken on a single page in such a fashion that any text precedes it in the flowing text area of the page, then a new page is forced following the end of this material. This case is similar to inline="unsplit" with the added requirement that a page break must be forced following the material under certain circumstances.
5. Order within float location is preserved even between material that floats and material that appears inline. It is the start of the inline or floating material that determines the order.
6. Broken inline material will not automatically keep out floats of the same float location. Restricting the floats that can interrupt the inline material must be done with the keep floatsout characteristic, as with any other inline material.
7. A float that gets placed inline is no longer under the control of the maximum size constraints that apply to either a single float location or all float locations cumulatively for the page. Thus an inline object that could have floated to a

float location associated with a page, flowing text area, or column will not be limited by any size constraints and will be allowed to fill entire pages.

j. It is possible for Page Type specifications to create sets of constraints that cannot be resolved. For example, specifying a Page Type of Same for more than one float per page can, depending on their sizes and other constraints, easily create an unresolvable situation. Also, having a float with Page Type of Next followed by a float of Page Type of Same will make it impossible to maintain the order of the floats. In such cases, the output system may either issue an error message or attempt resolution that does not satisfy all the constraints. The FOSI writer should avoid writing specifications that lead to such over-constrained situations.

k. This Output Specification says nothing about the details of how Page Types of Facing or Front/Back work. In particular, it does not describe how the output system should handle the case that a specification of Facing or Front/Back requires that material gets floated "backward" onto a previous page. FOSI writers should realize that Page Type specifications of Facing or Front/Back may not be supported by all output systems, or may have system dependent results with very different appearances.

l. If the float is associated with a Float location whose Float Type is Multi-reference, the float appears exactly once per page for any page from which this float was referenced one or more times. To be able to determine when two references are for the same multi-reference float, each reference to a multi-reference float must have an identifying name. For each float associated with a Float location whose Float Type is Multi-reference, the floating e-i-c instance's Float category's Multi-reference Name characteristic must specify an identifier that identifies this multi-reference float. (All multi-reference floats for which no Multi-reference Name is given are considered to be equivalent; that is, they are all treated as if they were given the same name.) The scope of the Multi-reference names is the same as the Scope of the multi-reference floats. Within the entire scope of a given multi-reference float, the content (the material being floated) is taken from the first reference using a given Multi-reference Name, and this content is used (and any given content is ignored) for all other references throughout the scope of that multi-reference float.

m. Within the current element's characteristic list, the Textbrk category's characteristics of startln and endln are both treated as if they were set "on" ("1"). Startcol has no meaning and is ignored. The characteristics startpg, newpgmdl, and pageid are relevant for the floating material (but have no effect on the material that does not float) as follows: if startpg is not off and newpgmdl=local, then the floating element begins on a new page and the pageset specified by the pageid is used for the duration of the floating element's content. This allows, for example, the floating of an element into a series of landscape pages. (A value of newpgmdl=global is treated as if newpgmdl=local when the e-i-c instance in question floats.)

n. Between individual floats within a single float location, the pre-/postspace specified in each floated element combines in the usual way with the pre-/postspace of adjacent floats.

B.4.4.19 Alignment group. Allows for placing potentially multi-line elements horizontally next to each other in the output data stream.

Characteristics - Definitions:	Values - Definitions:
Reference point - Vertical alignment point	List (Top, First, Last, Bottom, Middle)
Postspace - space under wrapped elements	Toggle

EXPLANATION:

a. When included in a charlist, algroup determines the relative position of the output element that follows the element specifying the algroup category provided that the margins and indents of the two elements are specified in such a way as to avoid overprinting. Alignment does not occur if the margins and indents are such that overprinting of text would occur (see note c). Note that more than two elements can be aligned horizontally using algroup on all of the elements (with the possible exception of the last one to be aligned).

b. The options for reference point are: top, first, last, bottom, and middle. Top refers to the ascender, cap-height, or font height of the first line (the precise determination of which depends on the font and may be system dependent), first refers to the baseline of the first line, last to the baseline of the last line, and bottom to the descender of the last line (which is again font and system dependent). For example, by setting the reference point of two adjacent paragraph-like structures to first, they will appear side-by-side aligned at their initial baselines. Middle implies that the reference point will be the vertical center of the lines of material. While an individual implementation of “vertical center” is system dependent, it is recommended that, as far as practical, this be defined as midway between the “first” and “last” reference points. The reference point of the element following that with the algroup specification is treated as “top” if the preceding element with the algroup specification has a reference point of “top” and “first” otherwise.

c. If the first of two structures uses the Alignment Group category but the left margin (as determined by its Left Indent) of the second structure (which does not have an Alignment Group category specification) is not large enough to allow for the width of the first structure (as determined by that structure’s Right Indent), then no horizontal alignment will take place (and the two structures would be positioned as if the Alignment Group category had not been specified). However, if the second structure’s First Line indent is large enough to allow for the width of the first structure, then horizontal alignment will take place (regardless of the non-first line left margin of the second structure). In this case, the second structure will use the value of its First Line indent for its left margin for as

many lines as necessary to clear the first structure, at which point the second structure's left margin will then revert to the value as specified by its Left Indent. This allows for some simple automatic wrapping of the right hand structure's text around the left hand structure.

d. An element without an algroup specification can be part of an alignment group only if the element immediately preceding it has an algroup specification and its margins and indents are such as to avoid overprinting with the preceding element. The first element following an alignment group that is not part of the alignment group is positioned below the alignment group using its margins and indents as usual in the case when no aligning is involved.

e. The vertical spacing between the aligned set of elements and the preceding element is determined by:

   1. Calculating the vertical spacing that would normally result between the preceding element and the first element of the horizontally aligned group without consideration of the algroup.
   2. Applying this vertical spacing between the preceding element and that element of the aligned group whose vertical extent most extends back toward the preceding element when all aligned elements are aligned as specified.

f. The vertical spacing between the aligned set of elements and the following element is determined by: (1) calculating the vertical spacing that would normally result between the last element of the horizontally aligned group and the following element without consideration of the algroup and then (2) applying this vertical spacing between the following element and, regardless of wrapping, that element of the aligned group whose vertical extent most extends forward toward the following element.

g. When the postspace toggle is set "on", the postspace, as provided in the charlist for the element, is applied before the adjacent group element can wrap underneath. When the toggle is set "off", no postspace is applied to the element before wrapping of the adjacent element occurs.

B.4.4.20 Suppress. Suppressing text from the output data stream.

Characteristics - Definitions:	Values - Definitions:
Suppress	Integer

EXPLANATION:

a. The integer value of the Suppress characteristic shall be one of 0, 1, 2, 3, 4, or 5. Each value determines a different level of suppression as defined in these notes.

b. To discuss the various levels of suppression, it is useful first to describe various classes of things that can be suppressed. An arbitrary element may contain character data, element content, or both. Furthermore, any element instance's

e-i-c's resolved charlist may contain various categories that contribute something to the output stream (for example, border, usetext, puttext, putgraph, ruling, and textbrk). The charlist may contain categories that do not contribute directly to the output stream but that do potentially have effects (such as modifying global FOSI counters) that go beyond the scope of the element (such as enumerate and savetext). The charlist may also contain categories with effects on the contained character data that are scoped by the element instance (for example, font, quadding, indent, and others).

c. The classes of things that can be suppressed follow:
    1. Class a, all categories of the current e-i-c that contribute something to the output stream (for example, border, usetext, puttext, putgraph, ruling, and textbrk).
    2. Class b, all categories of the current e-i-c that do not contribute directly to the output stream but that do potentially have effects that go beyond the scope of the element (for example, enumerate and savetext).
    3. Class c, the element content (the children) of the current element.
    4. Class d, character content of the element. (Note: all data referenced by an entity attribute in the source document that is to be inserted into the output stream—such as external graphics where composition is specified using the graphdesc section of the FOSI—are considered to be in the same class as the character data.)

d. The following discussion of the various levels of suppression will refer to the defined classes of things that can be suppressed by their letter a, b, c, and d. Table I summarizes the effect of each valid integer value of the Suppress characteristic in terms of these defined classes. Any non-zero value of suppression suppresses the character data (class d) for the element and all its descendants.
    1. A value of "0" means that this Suppress category has no effect on processing. In general, this means that no suppression is in effect (though, if an ancestor of the current element instance is already doing suppression, then that level of suppression cannot be affected by this or any other of its descendants).
    2. A non-zero value of suppression indicates some level of suppression as described. Note that once any (non-zero) level of suppression is commenced by any element in the document instance, that level of suppression remains in effect until that element is closed, and the level of suppression will not be altered by any of that element's descendants (regardless of what their e-i-cs do to the Suppress characteristic).
    3. A Suppress value of "1" means that the complete charlist of the current e-i-c will first be processed, but then all of that element's content (both character data and element content) will be suppressed. That is, the behavior is as if processing of the document instance "jumps" to the beginning of the corresponding end tag after completing the processing associated with the element's start tag. In terms of the defined classes, this means that no

effects of the current e-i-c (neither classes a nor b) are suppressed, but the element content (class c) and all character data are suppressed.

4. A Suppress value of “2” means that the complete charlist of the current e-i-c will be processed, as will all of its descendant element’s e-i-c’s, but all character data will be suppressed. In terms of the defined classes, this means that no effects of the current e-i-c (neither classes a nor b) are suppressed. Likewise, no effects of any descendant e-i-c’s (neither their class a nor b categories) are suppressed; however, all character data (of the current and all descendant elements) will be suppressed.
5. A Suppress value of “3” means that the complete charlist of the current e-i-c will be processed, as will the categories of its descendant element’s e-i-cs that do not contribute to the output stream, but all character data as well as the categories of its descendant element’s e-i-cs that do contribute to the output stream will be suppressed. This is similar to a value of “2” with the added suppression of class a categories for the descendant e-i-cs. In terms of the defined classes, this means that no effects of the current e-i-c (neither classes a nor b) are suppressed. For all descendant e-i-cs, class a categories are suppressed, but class b categories are not suppressed; furthermore all character data (of the current and all descendant elements) will be suppressed.
6. A Suppress value of “4” means that the charlist of the current e-i-c will be processed with the exception of the class that contributes to the output stream; furthermore all of that element’s content (both character data and element content) will be suppressed. This is similar to a value of “1”, with the added suppression of class a categories for the current e-i-c. In terms of the defined classes, this means that for the current e-i-c, categories in class a are suppressed, categories in class b are not suppressed, and the element content (class c) and all character data are suppressed.
7. A Suppress value of “5” means that the charlist of the current e-i-c will effectively be entirely suppressed; furthermore all of that element’s content (both character data and element content) will be suppressed. This is effectively equivalent to not having this element (and all its content) in the document instance at all. (Note that the charlist—including all its attspecs—will need to be examined to determine if the Suppress characteristic is given the value of “5” before any other charlist processing occurs, since if Suppress equals “5”, then none of the class a or b categories of the charlist have any effect.) In terms of the above defined classes, this means that, for the current e-i-c, all categories in both class a and b are suppressed, as well as all its element content (class c) and all character data.

APPENDIX B

TABLE B-1. Suppression mapping

Mapping of Suppress Values to Class Categories						
	Current Element			Current Element's Children [Class c]		
Class	a	b	d	a	b	d
Suppress Value						
0	-	-	-	-	-	-
1	-	-	S	S	S	S
2	-	-	S	-	-	S
3	-	-	S	S	-	S
4	S	-	S	S	S	S
5	S	S	S	S	S	S

Key:

a - All categories of the element-in-context that contribute something to the output stream (for example, border, usetext, puttext, putgraph, ruling, and textbrk).

b - All categories of the element-in-context that do not contribute directly to the output stream, but that do potentially have effects that go beyond the scope of the element (for example, enumerate as savetext).

c - The children (and by extension, all descendants) of the element.

d - Character content of the element.

S - Suppressed.

B.4.4.21 Boxing. The drawing of a box.

Characteristics - Definitions:	Values - Definitions:
Top Offset	Size/Distance
Bottom Offset	Size/Distance
Left Offset	Size/Distance
Right Offset	Size/Distance
Top Offset Origin - The method for determining the zero position for Top Offset.	List (Top, First)

Bottom Offset Origin - The method for determining the zero position for Bottom Offset.	List (Last, Bottom)
Side Offset Origin - The method for determining the zero position for Side Offset.	List (Text, Column, Content)
Left Gap - Additional space between boxed material and box's left side.	Size/Distance
Right Gap - Additional space between boxed material and box's right side.	Size/Distance
Thickness	Size/Distance
Top Rule Type	List (Blank, Single, Bold, Double, Triple, Dot, Dash)
Bottom Rule Type	List (Blank, Single, Bold, Double, Triple, Dot, Dash)
Left Rule Type	List (Blank, Single, Bold, Double, Triple, Dot, Dash)
Right Rule Type	List (Blank, Single, Bold, Double, Triple, Dot, Dash)
Outline Color - Color of the outline rules.	List (Black, White, Red, Orange, Yellow, Green, Blue, Violet, Brown, Gray)
Outline Screen - Shading of the outline rules.	Integer
Interior Color - Color of the area within the outline rules	List (Black, White, Red, Orange, Yellow, Green, Blue, Violet, Brown, Gray)
Interior Screen - Shading of the area within the outline rules.	Integer

EXPLANATION:

a. The integer value for Outline and Interior Screen is specified as a percentage. For example a value of "80" means 80% of black.

b. The offsets determine how far away from the boxed content area the rules will be drawn (positive values always mean "away" from the content).

c. Note that boxing causes the boxed material to become an unbreakable unit (the entire region will behave as if keep were enabled for it); therefore a boxed region will not break over columns. The boxing category is also permitted in subchars; that is, a box can be put around material generated by a Puttext or Usetext.

d. If startln="1" is in effect for the text being boxed, siderel of "text" means the width of the boxed content area is considered to be the full width to the margins currently in effect. Siderel of "col" means the width of the boxed content area is considered to be the full width of the current column (regardless of margins).

Siderel of “content” means the width of the boxed content area is considered to be the width taken by the widest extent of text or ruling within the boxed content. When startln="1" is in effect, the leftgap and rightgap characteristics are ignored.

e. If startln="1" is not in effect for the text being boxed, siderel is treated as equal to content and the box surrounds the in-line text. Note that the boxed text is unbreakable (that is, cannot break over lines) and must therefore fit on one line in the output. When startln="1" is not in effect, the leftgap and rightgap characteristics give an amount of space to add (inside the box) preceding and following the actual content being boxed. (Values of zero would mean the left and right rule would touch the boxed text; positive values are, in both cases, “away” from the text.)

f. A “bold” rule is a rule whose thickness is twice that of a single rule. A “double” rule is composed of two rules separated by a gap; the overall thickness of the double rule is equal to a bold rule (twice the thickness of a single rule). The thickness of each of the rules should be half the thickness of a single rule and that of the gap be the same as the thickness of a single rule.

B.4.4.22 Link. Characteristics for specifying links to other parts of the same or different documents

Characteristics - Definitions:	Values - Definitions:
System identifier - a specification of a location to which to link	String
Target document entity - a pointer to the document containing the link target	Pointer
Target ID - an ID reference to the link target	Name of ID in target document
End Target ID - an ID reference to an endpoint in the link target	Name of ID in target document
Link Type - indicates some particular link event behavior	List (Unspecified, GoTo, NewView)
Use Link - indicates which link definition characteristics to use	List (None, CurrDoc, SysId, TargDoc)
Use Target Id - indicates whether Target ID characteristic is used	Toggle
Use End Target Id - indicates whether End Target ID characteristic is used	Toggle

EXPLANATION:

a. The System identifier (SysId) is a string that specifies the link target. (If the System identifier specifies an SGML entity, the actual link target may be further specified via the Target ID and End Target ID characteristics.) The format and interpretation of the string is left to the output (presentation) system. The use of System identifier precludes the use of Target Document Entity.

b. The Target document entity (TargDocEnt) references (using SGML's entity referencing mechanism) the entity that is the target of the link. (If the Target document entity specifies an SGML entity, the actual link target may be further specified via the Target ID and End Target ID characteristics.) If this characteristic is left unspecified or uselink="currdoc", then the effect will be as though the Target document entity characteristic references the entire current document (not counting subdocs). The use of Target document entity precludes the use of System identifier.

c. The Target ID references an ID in the target document's ID name space. This reference points to the location to which to link. In particular, the beginning of the element identified by the Target ID shall be the target point. If the specified Target ID is not a valid ID in the name space of the document specified via the SysId or TargDocEnt characteristic (or if the target specified via the SysId or TargDocEnt characteristic is not an SGML entity), then the Target ID characteristic is ignored. If the Target ID characteristic is omitted or ignored, the actual target shall be the beginning of the target specified by the SysId or TargDocEnt characteristic.

d. The End Target ID references an ID in the target document's ID name space. This reference points to the location determining the end of the target. It is up to the presentation system to determine precisely how to interpret this, but the intent is that, upon reaching the end of the object identified by the End Target ID, the system will perform a "link back" to the location from which this target was linked to. If the specified End Target ID is not a valid ID in the name space of the document specified via the SysId or TargDocEnt characteristic (or if the target specified via the SysId or TargDocEnt characteristic is not an SGML entity), then the End Target ID characteristic is ignored. If the End Target ID characteristic is omitted or ignored, the effect shall be as if the End Target ID specified the end of the target specified via the SysId or TargDocEnt characteristic.

e. The target of any link (however specified) is interpreted by the output (presentation) system, and the system may treat this target as the final target or it may interpret it further as somehow indicating the final target. For example, when using a system with HyTime capabilities, a Target ID might point to a HyTime construct that is itself an indirect address to a different location.

f. The Link Type characteristic specifies some particular behavior with respect to the link event. If the Link Type is GoTo, the currently presented portion of the document becomes no longer presented and the target location becomes the

only portion now presented; that is, the display changes from the “linked from” to the “linked to” location, and no new display window is created. If the Link Type is NewView, the target location becomes presented in addition to that currently being presented; that is, a new window is created wherein the “linked to” location is displayed. A Link Type value of Unspecified merely implies that no specific semantic is given in this specification–see explanation note h. While this characteristic allows for some more interchangeable specifications, it still does not attempt to define the details of how a link event is implemented for any given system.

g. The link category gives specifications for the link that should happen when a link event is triggered for this e-i-c. It is left to the output (presentation) system to determine what causes a link to be triggered. The usual intent is that the element associated with the link category is “active” or “hot” or has an associated object (for example, a button) that is “hot” and that a mouse click or similar action on the “hot” region would trigger the link event.

h. The precise behavior associated with the “link to” event is left to the output (presentation) system. The Link Type characteristic (see explanation note f.) can be used to indicate a preference for one of the predefined behaviors, for example, updating the current display window to show the target location or creating a new window to show the target location. However, the details of how this works are left to the output system. Furthermore, other behaviors may also be supported by the output system. An output (presentation) system may choose a specific interpretation for any link request based on various factors about the location linked from, the location linked to, or details of the specification of the target (via either the System Id or the Target Document Entity and ID).

i. The effect of the link category holds for the entire subtree of the current element instance in the output tree: that is, the element whose e-i-c contains the link specification and all of its descendants as well as for all material generated by ruling, puttext, putgraph, or usetext associated with the element or its descendants, but not including any material that may float out of the region bounded by the element containing the link specification. (A link specification attached to an element that itself floats or is contained within a region that floats remains associated with the element on which it occurs.) Link specifications can be nested (treated like a stack). The ending of a nested Link region causes the effect of the link region that was in effect before the nested region started to be once more in effect.

j. The Use Link (UseLink) characteristic indicates which of the various link characteristics should be used and which should be ignored. Its values have the following meanings:

    1. The value “none” means that this entire link category is ignored as if it were not present in the charlist for this e-i-c.

2. The value “currdoc” means that both the System identifier and the Target document entity characteristics are ignored and the entire current document becomes the target entity.
3. The value “sysid” means the System identifier is used to locate the target entity; the target document entity characteristic is ignored.
4. The value “targdoc” means the Target document entity characteristic is used (via standard SGML resolution of the associated external identifier) to locate the target entity; the System identifier characteristic is ignored.

k. Regardless of how the target entity is determined, whether the Target ID or the End target ID characteristics are used or ignored depends on the settings of the UseTargId and UseEndTargId characteristics. If the UseTargId toggle is true “on”, the Target ID characteristic shall be used, and if the UseTargId toggle is false “off”, the Target ID characteristic shall be ignored. Similarly, If the UseEndTargId toggle is true “on”, the End Target ID characteristic shall be used, and if the UseEndTargId toggle is false “off”, the End Target ID characteristic shall be ignored.

B.4.4.23 Link process. Characteristics for indicating external processes to execute in association with link events (see also the Link category).

Characteristics - Definitions:	Values - Definitions:
Link out process - a pointer to a process to execute when a link event is triggered	Pointer
Execute link out process - indicates whether the link out process characteristic should be considered	Toggle
Link out construction rule - a string that is passed to the link out process	String
Link in process - a pointer to a process to execute when a “linked to” event is triggered	Pointer
Execute link in process - indicates whether the link in process characteristic should be considered	Toggle
Link in construction rule - a string that is passed to the link in process	String

EXPLANATION:

a. The Process characteristics (both the Link out process and Link in process) reference (using SGML's entity referencing mechanism) an entity that specifies a process to be executed. The format and interpretation of this entity is left to the output (presentation) system. The entity may simply be an internal text entity that gives a system command, it may be an executable file, or it may be an indirect indication of what process should get executed.

b. The Link Process category gives specifications for the link that should happen when a link event is triggered for this element-in-context. The same event(s) that triggers the link (see B.4.4.22) shall trigger the Link out process, and the process shall be executed prior to the link being traversed. Any event may trigger the Link in process, but it is expected to be triggered by the process of being linked to (and perhaps by the element instance's first being displayed upon the presentation device).

c. As indicated in the description of the link category, an output (presentation) system may choose a specific interpretation for any link request based on various factors; the Link out process could potentially be used in helping to determine the interpretation.

d. The Execute link out process and Execute link in process toggles indicate respectively whether the Link out process or Link in process characteristics should be considered. If the toggle is "off" ("no", "zero"), then the respective characteristic is ignored regardless of its setting. If the toggle is "on" ("yes", "nonzero"), then the respective characteristic is inspected and its value, if any, is used when appropriately triggered.

e. The Link out construction rule and Link in construction rule are Construction rules (see B.3.4.10) that can be used to pass information to the Link out process and Link in process. It is up to each process how it will interpret this information.

B.4.4.24 Reset. The resetting of counters and string variables declared in the Resource Description (via the Counter or String constructs).

Characteristics - Definitions:	Values - Definitions:
Reset Control - The list of unique Counter Ids or Savetext Names to be reset to the initialized value of the corresponding variable or Float Location Ids whose contents should be flushed.	String

EXPLANATION:

a. A counter is reset to its initialized value (as defined in the Counter specification for this Counter ID in the Resource Description) when it appears in the Reset

Control list associated with an e-i-c and that e-i-c is encountered in the source document.

b. A string variable (Savetext Name) is reset to its initialized value (as defined in the String specification for this Savetext Name in the Resource Description) when it appears in the Reset Control list associated with an e-i-c and that e-i-c is encountered in the source document.

c. The contents of all Float Locations whose ID are in the Reset Control list is “flushed”. For Float Type of Once and Multi-Reference, flushing means the floats are forced out into the output stream preceding any material the current e-i-c would supply to the output stream even if this would violate the floating constraints. For Float Type of Repeat at Page Break, the float locations are emptied without contributing anything further to the output stream.

d. The Reset Control string is interpreted as a list of variable names. These names are generally name tokens except that they may be prefaced with “global:” to refer to the globally scoped variable of the given name from within a FOSI module (ospackage). The Reset Control string is treated in a case-insensitive manner when considering parts of it as potential variable names.

B.4.4.25 Enumeration. The ordering of like elements. The ordering is a property of individual elements, which may be combined to form a compound number. Note that counters referenced by this category must be declared using the Counter construct that appears in the Resource Description.

Characteristics - Definitions:	Values - Definitions:
Increment - The number by which to increment the counter for the element before using it this time.	Integer
Counter ID - A unique name assigned to a counter.	String
Set Value - A toggle indicating whether the counter is reset to zero before the Increment value is applied.	Toggle

EXPLANATION:

a. The counter associated with an e-i-c is incremented by the Increment each time that e-i-c is encountered in the source document.

b. If the Set Value toggle is enabled, the counter specified by the Counter ID is reset to zero before any increment is applied. (This has the effect of setting the Counter to the Increment value.)

c. Compound numbers are specified through the Source of the Usetext category or the Construction Rule.

d. Specifying Enumeration only sets up the characteristics for a counter associated with an element. To specify the position of the counter in the output, the counter must be used in the Source of a Usetext specification (either directly by its Counter ID or through its inclusion in a Construction Rule of a Savetext).

e. See B.3.8 for a discussion on the order of processing of multiple Savetext's, Usetext's, and Enumerate's.

f. The Counter ID string is interpreted as the name of a counter FOSI variable. This name is generally a name token except that it may be prefaced with "global:" to refer to the globally scoped variable of the given name from within a FOSI module (ospackage). The Counter ID string is treated in a case-insensitive manner.

B.4.4.26 Ruling. The drawing of a horizontal rule.

Characteristics - Definitions:	Values - Definitions:
Thickness	Size/Distance
Length Type - The method for prescribing the length of the rule.	List (Specific, Relative)
Specific Length - The measure of the rule.	Size/Distance
Relative Length - The length of the rule relative to a layout area.	List (Column Width, Text Width)
Vertical Offset - A vertical adjustment from the baseline.	Size/Distance
Placement - Placement of the rule before or after the element content.	List (Before, After)
Rule color - Color of the rule.	List (Black, White, Red, Orange, Yellow, Green, Blue, Violet, Brown, Gray)
Rule Percent - Percent giving color density of the rule.	Integer
Type	List (Blank, Single, Bold, Double, Triple, Dot, Dash)

EXPLANATION:

a. Either a Specific Length or a Relative Length can be specified. A Specific Length indicates a known measure. A Relative Length indicates the rule should automatically be drawn to the full width of the Column Area or to the "Text Width" which is the column width after indent values are applied.

b. If Textbrk's Start Line is not in effect, a Specific Length must be given, and the rule will be drawn in-line with the surrounding text. It is an error for the Specific Length to be greater than the available space.

c. A positive Vertical Offset raises the rule above the baseline, while a negative value lowers the rule below the baseline.

d. The Placement characteristic indicates whether to draw the rule before or after the element content.

e. The integer value for Rule Percent is specified as a percentage. For example, a value of "80" means 80% of black.

f. When Ruling is specified multiple times, or is specified along with Puttext, Putgraph, and Usetext, the order of the generated data in the output stream should appear in the order in which the characteristics are specified for the e-i-c (both before and after the content).

B.4.4.27 Puttext. Describes system-generated text that appears in the output data stream.

Characteristics - Definitions:	Values - Definitions:
Puttext Literal - The string that appears.	String
Placement - Whether to place the Puttext literal before or after the content of the element.	List (Before, After)

EXPLANATION:

a. This characteristic is used to specify a string of text that is generated by the system and is put into the text stream. For example, the system may automatically generate the "This Page Intentionally Left Blank" string for use on blank pages.

b. All characters appearing in the Puttext Literal String are reproduced. Should the string require a quote, the alternate quote should be used to enclose the Literal. For example, if a double quote (") is used within the string, the single quote (') should be used to enclose the Literal, and vice versa.

c. When Puttext is specified multiple times, or is specified along with Ruling, Putgraph, and Usetext, the order of the generated data in the output stream should appear in the order in which the characteristics are specified for the e-i-c (both before and after the content).

d. The Puttext content is placed immediately before or after the contents of the element (properly mixed with other Puttext, Putgraph, or Usetext generated data). Any Prespace or Postspace of the element or any effects of other characteristics of

the element (such as those due to Textbrk, or Keeps) occur previous to “Before” generated text and subsequent to “After” generated text.

B.4.4.28 Putgraph. Describes system-generated graphics that appear in the output data stream.

Characteristics - Definitions:	Values - Definitions:
Graphic Name - A reference to the graphic that is to appear.	Pointer
Width - The width of the graphic.	Size/Distance
Depth - The depth of the graphic.	Size/Distance
Placement - Whether to place the graphic before or after the content of the element.	List (Before, After)
Scale to Fit	Toggle
Horizontal Scaling	Integer
Vertical Scaling	Integer
Horizontal Offset	Size/Distance
Vertical Offset	Size/Distance
Rotation	Toggle

EXPLANATION:

a. This characteristic is used to specify graphics that are placed within running text, for example a symbol. For purposes of positioning, the composition system should treat the graphic as a single character.
b. When Putgraph is specified multiple times, or is specified along with Ruling, Puttext, and Usetext, the order of the generated data in the output stream should appear in the order in which the characteristics are specified for the e-i-c (both before and after the content).
c. The Putgraph content is placed immediately before or after the contents of the element (properly mixed with other Puttext, Putgraph, or Usetext generated data); any Prespace or Postspace of the element or any effects of other characteristics of the element (such as those due to Textbrk, or Keeps) occur previous to “Before” generated text and subsequent to “After” generated text.
d. The Width and Depth characteristics specify a bounding box, similar to the reproduction area Graphic characteristics, which represent the amount of space reserved on the presentation media for the graphic.

e. If either the Width or Depth characteristic is zero, then it is assumed that a graphic design width and design depth are available from the graphic data. If both width and depth are zero, the graphic design width and design depth are used to determine the bounding box dimensions. If only one of the width or depth characteristics is non-zero, then the other one will be determined using the proportion of graphic design width to graphic design depth.

f. If both width and depth are set to zero and the incoming graphic does not provide these precise measurements, the depth will default to the current font size and the width will be determined by using the rule in the preceding note.

g. The Scale to Fit characteristic allows the graphic to be scaled as needed to fit the size of the bounding box. Horizontal and Vertical Scaling characteristics take precedence when they have values other than "0". If both width and depth characteristics are zero, then the Scale to Fit characteristic has no effect, and the graphic design size will be used subject to any Scaling.

h. If either the width or depth characteristic is non-zero, and Scale to Fit is not in effect, the graphic image may be a different size than the bounding box. Whenever this is the case, the graphic image is aligned with the lower left corner of the bounding box, prior to shifting per the Horizontal or Vertical Offset.

i. Horizontal Offset and Vertical Offset are not subject to Scaling. These do not move the bounding box; they shift the graphic image within the bounding box.

B.4.4.29 Savetext. Describes text content to be saved for use elsewhere.

Characteristics - Definitions:	Values - Definitions:
Savetext Name - A unique name for the saved text.	String
Construction Rule - Rules for specifying the construction of saved text. The string may contain an indicator for the element content, literals, and unique names of other saved text and counters.	String

Placement - Whether to save the text before or after the content of the element is processed.	List (Before,After)
Append - Indicates whether the text to be saved should replace text already saved to this Savetext Name or append itself onto the text currently saved to this Savetext Name.	Toggle

EXPLANATION:

a. This characteristic is used specifically for saving portions of text for use by the Usetext characteristic for other purposes. For example, it might be used to save section titles for use in a Table of Contents.

b. See B.3.4.10 for a discussion of the Construction Rule Syntax.

c. See B.3.8 for a discussion on the order of processing of multiple Savetexts, Usetexts, and Enumerates.

d. Ordinarily when a subsequent Savetext is done with the same Savetext Name as a previous Savetext, the subsequent one replaces the previous one. This is the behavior when the Append characteristic has the value “off” (the default). However, if the value of the Append characteristic is “on”, the Savetext’s Conrule value would be “appended” to the text already saved to this Savetext Name. Using the Append characteristic, one can accumulate text from the document to aid in the production of tables of contents and similar constructs.

e. The Savetext Name string is interpreted as the name of a textid FOSI variable. This name is generally a name token except that it may be prefaced with “global:” to refer to the globally scoped variable of the given name from within a FOSI module (ospackage). The Savetext Name string is treated in a case-insensitive manner.

B.4.4.30 Usetext. Describes what to do with text saved from some part of the document.

Characteristics - Definitions:	Values - Definitions:
Source - Reference to saved data.	String
Placement - Whether to place the text before or after the content of the element.	List (Before, After)

Usage Rule - Describes what to do with the saved text.	Integer
Usage Parameter - Modifies the operation of the specified Usage Rule	String

EXPLANATION:

a. This category is used to output saved data. Specifically, it uses data saved with the Savetext, Enumerate, and Character Fill characteristics. Saved data is identified in the Source by unique names assigned to the saved data. A Usage Rule may be applied to the data before placing the result in the output text stream.

b. The syntax for the Source is the same as that for the Savetext Construction Rule and is described in detail in B.3.4.10. This allows for specification of multiple pieces of saved data in conjunction with literal text.

c. When Usetext is specified multiple times, or is specified along with Ruling, Puttext, and Putgraph, the generated data in the output stream should appear in the order in which the characteristics are specified for the e-i-c (both before and after the content).

d. The Usetext content is placed immediately before or after the contents of the element (properly mixed with other Puttext, Putgraph, or Usetext generated data). Any Prespace or Postspace of the element or any effects of other characteristics of the element (such as those due to Textbreak, or Keeps) occur previous to "Before" generated text and subsequent to "After" generated text.

e. See B.3.8 for a discussion on the order of processing of multiple Savetexts, Usetexts, and Enumerates.

f. If any userules are used which are not supported by this specification, they will be specified by a negative integer.

g. The Usage Parameter characteristic is used to provide parameters to the specified userule. This characteristic is ignored unless a non-zero userule is specified. The form of this characteristic is the same as that of the Puttext Literal characteristic. The syntax and semantics of the string specified for this parameter are determined by the specific userule with which it is used.

B.4.5 COMPOSITION — TABLES. This section describes the characteristics necessary for the composition of tables. Tables are treated separately in this specification because they have unique formatting characteristics. Tables are important in technical documents because they contain and present a large amount of data in a format that shows relationships among the data. The characteristics for tables described allow for robust and discretionary access and manipulation of data contained within tables, and facilitate exchange with, and use within, databases.

Within a FOSI, the table description is used to describe the organization and formatting of an actual table, that is, data organized into a two-dimensional grid. Any associated information, such as title, is described in the style description.

B.4.5.1 Tables as graphics. While it is recognized that tables are often created and treated as graphic illustrations, it is essential for automated processing of source data that the markup of tabular data be developed in such a way that allows the elements within a table to be mapped to the FOSI table description and that automated publishing systems be able to manipulate and reproduce tabular data through the use of a FOSI. Treating tables as graphics is not precluded, but should be carefully evaluated against requirements for information use. Note that treating a table as a graphic will likely make access of information contained in the table for uses other than display more difficult. This section deals only with tabular data appropriately tagged.

B.4.5.2 The structure of tables. Tables have two structures. The source structure describes the organization of table data within the source document. The output structure describes how that data actually appears in a two-dimensional format in the output. The purpose of the FOSI entry for a table is to describe how those two structures are related.

B.4.5.2.1 Source structure. The source structure of a table is defined in the source DTD. This structure should reflect the logical organization of the data as it relates to its purpose. Typically, however, this structure also reflects the two-dimensional output structure. The following is the source structure for a table as defined in MIL-HDBK-28001:

Table

Title
Table Groups

Column Specifications
Table Head

Column Specifications
Rows

Entries
Entry Tables

Table Foot

Column Specifications
Rows

Entries
Entry Tables

Table Body

Rows

Entries
Entry Tables

Note that this source structure is “generic,” that is, the elements are general and do not relate to any specific type of information that might go into the table. Other source structures for tables could be defined that more specifically detail the table content. For example, a source structure for a Special Tools List table could look like this:

Special Tools List

Tools

Description
Part Number
Reference

In this case, the elements in the source structure are more closely related to the data in the table.

B.4.5.2.2 Output structure and table objects. The output structure of a table is defined in this appendix. A table is a rectangular two-dimensional grid, with non-uniform measures, fixed horizontally and content-determined vertically. The objects within a table are table subset groups, table subsets, columns, rows, and cells. A table is the entire rectangle that takes up space in the Flowing Text Area. A table subset group is a set of table subsets containing an optional heading subset, an optional footing subset, and one or more body subsets. A table subset is a set of contiguous rows within a table of which there are three types; header, footer, and body. There is no space between table subset groups or table subsets in a table.

The rows of heading or footing subsets are repeated above and below (respectively) each contiguous (unbroken), block of rows from any of the table body subsets in the same table subset group. Thus, if the table body subsets breaks across pages or columns, the heading and footing subsets for that table subset group are repeated.

Table subsets may have different numbers of columns, but must be the same width as the table. A column is a vertical collection of cells. A column has some specified width and is the depth of the table subset. A row is a horizontal collection of cells. The width of a row is the width of the table and the depth of a row is the depth of the deepest cell. A cell is the intersection of a column and row and forms the basic area into which table content is placed. Table objects have the following hierarchy:

Table

Table Subset Groups (1 or more)

Column specifications (0 or more)
Table Subsets (1 or more)

Column specifications (0 or more)
Rows (1 or more)

Cells (1 or more)

In other words, tables are made up of table subset groups, (typically a single subset group), which contain table subsets, which are made up of columns and rows, which intersect to form cells.

B.4.5.3 Mapping source structure to output structure. Through the Table Description of a FOSI, the source structure of a table in the source DTD is mapped into the output structure defined in this appendix. Each source element is mapped into one or more table object(s). For example, the table source structure in MIL-HDBK-28001 has the following mapping:

Table	Table (tabatts)
Tgroup	Table Subset Group (tgroupatts)
Colspec	Column (colatts)
Spanspec	Column (colatts)
Thead	Table Subset (subsetatts)
Tfoot	Table Subset (subsetatts)
Tbody	Table Subset (subsetatts)
Row	Row (rowatts)
Entry	Cell (cellatts)
Entrytbl	Table Subset Group (tgroupatts)

Notice that the table title in the source structure does not have a corresponding object in the output structure. Formatting of the table title is specified in the style description.

Other table structures may be mapped to the table output structure differently. For example, the source structure for a Special Tools List made up of Description, Part Number, and Reference might be represented with the following simple DTD fragment:

```
<!ELEMENT tools - - (desc, partnum, ref)+>
<!ELEMENT (desc, partnum, ref) - o (#PCDATA)>
```

Here, “tools” is mapped to a table in the FOSI, and “desc”, “partnum”, and “ref” are mapped to cells. To fully specify the output of the table, however, may require specifying characteristics for columns and rows, which do not have corresponding elements in the source.

This table, including an implied heading, may be mapped to the table object output structure as follows :

Tools	Table
	Table Subset Group
	Columns (3)
	Table Subset (heading)
	Row
	Cells (3)
	Table Subset (body)
Desc	Row
	Cell
Partnum	Cell
Ref	Cell

Note that each element may be mapped to one or more objects in the table object hierarchy, with specification of appropriate characteristics for each. This mapping is accomplished by specifying the table output objects and their characteristics in the charlist of the e-i-c of each relevant element. Each table output object has a category in the output specification, with associated characteristics.

When a source element is mapped into more than one object in the table object hierarchy, the range of table objects into which the source element is mapped must be contiguous in the table object hierarchy. It is an error for an element to be mapped to a discontinuous range of elements. Thus, an error would occur if a FOSI attempted to map an element into a table subset and a cell, but not into a row.

The content of a source element must be contained within the lowest level table object specified in the charlist for that source element. Because of this (with one exception), an element can only be mapped to table objects which are lower in the hierarchy than any table objects a parent element is mapped into. The one exception is that is allowable to map an element into the tgroupatts table object even if a parent of the element has been mapped into a table object lower in the hierarchy. One reason for this exception is to allow for the specification of an entry table (entrytbl) which is used for a table within a table. For example, an error would occur if a source element were mapped into a cell, and one of its sub-elements were mapped into a table subset but it would not be an error if an element was mapped into a row and one of its subelements was mapped into a table group.

Each table object shall be mapped via the FOSI into one or more of the source table elements.

To place text or graphics directly from the FOSI into the output table objects, as shown in the described mapping, use the appropriate table object categories in the subchars specification of a usetext, puttext, or putgraph. One use for this technique is to generate a table head for a content table such as the Special Tools List Table.

The e-i-c for source elements to be mapped into table objects may use the att and specval or fillval constructs to create table specifications which are dependent on attributes of the source document elements. This supports the implementation of named table and table group styles.

B.4.5.4 Text flow rules of tables. Text flows within tables from left to right and from top to bottom. That is, the first column of the first row (the first cell) is filled with the necessary text; then the second column of the first row (the second cell) is filled. After the last column of the first row is filled, text flows to the first column of the second row, and so on.

Columns are implicitly numbered starting with 1 at the leftmost column and incrementing by 1 to the right. Rows in each table subset are implicitly numbered starting with 1 at the top row and incrementing by 1 to the bottom. A cell is uniquely identified by its position within a row and column.

Spanning is the creation of cells that are larger than a single grid location. Span width indicates the number of columns covered horizontally by a cell. Span depth indicates the number of rows covered vertically by a cell. Spanning occurs in the direction of the flow of cell filling, that is, for horizontal spanning, to the right of the cell where spanning is specified, and for vertical spanning, towards the bottom from the cell where spanning is specified. A spanned cell is considered to be part of the first row and first column in which it occurs.

The source markup for a table entry can explicitly specify the name of the column in which the cell content is to occur. When this information is not supplied, a flow rule is used to determine the cell location. Entries are placed in the “current column,” which if not previously specified is column 1. After a column is filled, the current column becomes the column with the number “current column number + span width” (where no spanning means the span width is “1”).

When fewer entries occur than cells in a row, the remaining cells are treated as if they were each filled by an entry with empty content; that is, row and column separators appear for those cells that would have had them had the cell been filled.

B.4.5.5 Specifying the style of a table. Two aspects to specifying the style of a table are geometric and text composition. The geometric aspect includes the number of columns, rulings, and margins. The text composition aspect includes font, positioning of text within cells, and generally those characteristics that can be applied to text. Both of these kinds of characteristics can be specified in the FOSI. Special table characteristics are provided to control the style of the table itself. Composition characteristics are used to specify the style of the content.

B.4.5.6 Overriding FOSI values through the source markup. For any particular source DTD (such as the Example DTD in MIL-HDBK-28001), the values of many source attributes may map directly to Table characteristic values. Where this occurs, source attribute values can override any value supplied in the corresponding Table characteristic. A FOSI entry may be constructed such that values are supplied for characteristics but any values supplied

in the source override the specified value by using the Fillval construct. Authors need not specify all the attributes available to them. The attributes exist to allow authors the necessary freedom to create an exception to a particular Table Style. For example, an author may wish to specify that a particular column should have its contents center-aligned when referencing a Table Style that specifies that all columns are left-aligned. The author then needs only to specify a single attribute to get the desired result, as the "center" value the author specifies for the "align" attribute can override the "left" value specified in the FOSI. The "left" value remains in the FOSI so that the exception specified by the author remains the exception it is supposed to be.

B.4.5.7 Composition and table characteristics. There are composition and table characteristics which overlap such as quad characteristic of the quadding category and the halign characteristic of the stdcellatts category. With several exceptions these composition characteristics override the table characteristics already set up for the table objects. One exception is with indents. Indents specified for the content are additive to the margins established for the cell. The other exceptions involve composition characteristics which do not apply within tables.

The following Composition Characteristics do NOT apply within tables: Highlight Reverse, Background Color and Background Screen (cell Reverse, Color, and Shading take precedence); Keeps (row breaking takes precedence); Vertical Justification; Textbreak Start Column, Start Page, Page Model ID, and New Page Model; Span (cell spanning takes precedence).

While it may be sufficient to specify the formatting of simple text (#PCDATA) within a cell by supplying Cell Characteristics and Composition Characteristics for the cell, there may be a need to specify additional Composition Characteristics when the cell contains more complex element content. For example, lists within table entries may differ from lists in flowing text. These e-i-cs (that are children of the source table "entry") must be included in the Style Description of the FOSI and the appropriate Composition Characteristics specified.

B.4.5.8 Scope of table characteristics. In general, there is a unique set of characteristics for each table object. In addition, there is a set of "Standard Cell Characteristics" that control the characteristics of a cell but can be specified on any table object. These characteristics include column and row separators, margins, and alignment. When specified on a table object, these characteristics apply to all the cells within the scope of that object. For example, when Standard Cell Characteristics are specified on a column, those values apply to all the cells in the column. When specified on more than one table object, the values for objects "lower" in the hierarchy override the others for the scope of that object; that is, the order of precedence is cell, row, column, table subset, table subset group, and table. Note that this "table object specification inheritance" is different from the "source document characteristic inheritance" feature available elsewhere in a FOSI.

The table description categories (tabatts, tgroupatts, subsetatts, colatts, rowatts, cellatts, stdcellatts) in the charlist are intended only for the specification of table output object characteristics. The categories may be used in environments and charsubsets in the Style Description section of a FOSI to define environments and charsubsets which may be used in the Table Description section of the FOSI as part of a table description. The use of any

of these categories in an e-i-c in the Style Description section of a FOSI, either directly or by reference to an environment or charsubset, is an error.

B.4.5.9 Table characteristics. The characteristics listed below are unique to tables, and may be used in conjunction with Composition Characteristics to fully specify the output of tables.

B.4.5.9.1 Table element characteristics (tabatts). Characteristics that apply to the table as a whole.

Characteristics - Definitions:	Values - Definitions:
Width Type	List (Specific, Relative)
Specific Width	Size/Distance
Relative Width	INTEGER (percentage)
Frame	List (All, Top, Bottom, Top & Bottom, Sides, None)
Frame Thickness	Size/Distance
Frame Style	List (Blank, Single, Bold, Double, Triple, Dot, Dash)
Continue With Row Separator	Toggle

EXPLANATION:

a. FOSIs have a minimum of one Table Style specified as a result of supplying a default value for each Table characteristic.

b. Widthtype indicates whether the width of the table is a particular specified value or is relative to the column or page in which it is placed. When “specific” is specified, a value should be supplied for the Specific Width characteristic. When “relative” is specified, a value for the Relative Width characteristic should be supplied to indicate the percentage of the column or page the table is to fill.

c. Frame specifies on which side(s) of the table the frame is to appear.

d. The Frame Thickness is measured from the outside edge of the Table boundary inwards. Notice that this differs from the other rule separators.

e. Framestyle specifies how many lines will be used to create the frame. The total width of the frame will be the value specified in frame thickness. For those frames with more than one line the space between the lines will be equal to the thickness of the lines. Framestyles of bold, dotted and dashed will be composed of a single line.

f. When the Continue With Row Separator characteristic is “on”, a Table Row Separator appears under the last Table Row on a Column or Page and above the first Table Row on the succeeding Column or Page. When “off”, both Row Separators are omitted.

B.4.5.9.2 Standard cell characteristics (stdcellatts). Characteristics that apply to cells, which may be specified on any table object and apply to the cells within the scope of that object.

APPENDIX B

Characteristics - Definitions:	Values - Definitions:
Column Separator On	Toggle
Row Separator On	Toggle
Column Separator Width	Size/Distance
Row Separator Width	Size/Distance
Column Separator Style	List (Blank, Single, Bold, Double, Triple, Dot, Dash)
Row Separator Style	List (Blank, Single, Bold, Double, Triple, Dot, Dash)
Left Margin	Size/Distance
Right Margin	Size/Distance
Top Margin	Size/Distance
Bottom Margin	Size/Distance
Horizontal Alignment	List (Right - Flush Right & Ragged Left, Left - Flush Left & Ragged Right, Center - Centered, Justify - Flush Left & Flush Right, Char - A character to align to the left of)
Vertical Alignment	List (Top, Middle, Bottom)
Alignment Character	String
Alignment Character Offset	Integer (percentage)
Reverse	Toggle
Background Color	List (Black | White | Red | Orange | Yellow | Green | Blue | Violet | Brown | Gray)
Shading	Percentage
Rotation	Toggle
Text Width	Size/Distance
Break Row	Toggle

EXPLANATION:

a. When Column or Row separators are turned “on”, they have no effect on the Table Frame. When Column or Row separators are turned “on” and Frame is turned “off”, no rule appears around the outside edges of the table.

b. Column and Row separators appear centered over the right boundary and bottom boundary respectively. Thus they always appear between columns or rows. The separators shall appear centered between two columns or rows, such that when a column separator is 1 point thick, .5 points impinge on the width of the column

to the left of the separator and .5 points impinge on the width of the column to the right of the separator.

c. The values for row and column margins for a cell must be greater than half the thickness of any column or row separators that apply to that cell.

d. Any highlighting characteristics apply to the full cell area, extending through the margins to the cell separators.

e. Leftmar, rightmar, topmar and botmar are used to specify the left, right, top and bottom margins respectively of placement of the table cell content relative to the table cell boundaries.

f. Halign and valign specify how the cell content is to be placed relative to the table cell boundaries. A horizontal alignment of char causes the table cell content to be horizontally aligned relative to a specific character.

g. Alignment character is used to specify the character to be used for a horizontal alignment of “char”. The value of this characteristic is the single alignment character; the first occurrence of which, in any given entry, will be the alignment point for the entry. Entries not containing any occurrence of this character will be right-aligned to the left of the alignment character offset position (so that the rightmost character would be in the same position as it would if the entry had a single occurrence of the alignment character appended to the current content of the entry). If the value of this characteristic is represented by an entity reference (for example, SDATA character entity reference), the result is output system dependent.

h. Alignment character offset (charoff) determines the distance the cell content should be offset from the horizontal alignment character. The value of this characteristic is the percent of the width of the current column (or span of columns) to the left of the (left edge of the) alignment character.

i. For shading, “0” is white, “100” is full color saturation.

j. When the value for Rotation is “off”, the contents of the cell is placed in the same orientation as the Table. When the value is “on”, the contents of the cell is rotated 90 degrees counterclockwise.

k. The Textwidth characteristic is used only when Rotation is enabled. Textwidth specifies the width to which the text is formatted. When unspecified, Textwidth is the column width minus the left and right cell margins.

l. For cells with Rotation specified, the cell contents is formatted to the width specified by Textwidth using any applicable Composition Characteristics; the cell contents is rotated; then cell characteristics are applied.

m. When tables are allowed to continue across column or page boundaries, they shall normally break between rows. When the Break Row characteristic is turned “on” for all cells in a row, then a break across a column or page boundary may occur within that row.

n. No row with cells containing any graphic or rotated text can be split.

B.4.5.9.2.1 Table subset group characteristics (tgroupatts). Characteristics that apply specifically to Table Subset Groups.

Characteristics - Definitions:	Values - Definitions:
Number of Columns	Integer

EXPLANATION:

a. Cols specifies the number of columns in the table subset group.

B.4.5.9.2.2 Table subset characteristics (subsetatts). Characteristics that apply specifically to Table Subsets.

Characteristics - Definitions:	Values - Definitions:
Number of Columns	Integer
Keep - Keep the whole subset together within the specified boundary.	List (Column, Page, None)
Subset Type	List (Heading, Footing, Body)

EXPLANATION:

a. Cols specifies the number of columns in the table subset group.
b. The keep characteristic indicates the boundaries across which the subset cannot break, either a column or page boundary. Specifying “column” implies also keeping within the page.
c. The subset type characteristic indicates to which of the three table subset types to map this element.

B.4.5.9.3 Column characteristics (colatts). Characteristics that apply specifically to Table Columns.

Characteristics - Definitions:	Values - Definitions:
Column Width	String
Column Number	String
Column Name	String
Span Name	String
Start Column Name	String
End Column Name	String

EXPLANATION:

a. Column characteristics are used either to associate a column width and optional column name with an implicitly or explicitly numbered column or to associate the starting and ending column names with a span name. Either Column Width with Column Number and optional Column Name should be specified or Span Name, Start Column Name, and End Column Name should be specified. The width of a span is the sum of the widths of its contained columns.

b. The value of Column Number refers to the number of the column in the output structure. It can either be specified in the FOSI or obtained from the source. In a FOSI, a number refers to the implicit column number of a column in the output structure. The keyword "#LAST" is used to refer to the last (rightmost) column regardless of its implicit number. The keyword "#DEFAULT" can be used to specify a set of characteristics to be used when no column characteristics are specified for a column, either explicitly in a FOSI or obtained from the source.

   The value of Column Number can also be obtained from an attribute value specified for the source element that is being mapped to a column (via a Fillval). When this attribute has no value specified in the source, the Column Number is derived as one greater than the last Column Number specified (derived or explicitly) within the same table group. If no Column Number has yet been specified for the current table group, Column Number is 1.

   For example, suppose the "colspec" element is mapped to a column and its "colnum" attribute indicates the column number. If the first "colspec" element has no value for "colnum" specified, Column Number is assumed to be 1. If the next "colspec" element's "colnum" attribute has the value "3", Column Number would be "3" and the column characteristics for column 2 would have to be derived from a column characteristic using the #DEFAULT keyword. If the next "colspec" element had no value supplied for its "colnum" attribute, Column Number would be 4, and so on.

   The column width is specified either as a fixed measure or in terms of units of proportional measure with optional additional fixed measure. Fixed measure is specified as any other width in the FOSI: as a size/distance specification. Proportional measure is specified by an integer followed by "*". The unit of proportional measure is calculated by subtracting the sum of all fixed measures for all columns in a table from the width and dividing the remainder by the sum of the number of proportional units for all columns in the table. For example, the width specification "5*" would indicate five proportional units, while "5*3pt" indicates five proportional units plus 3 points.

c. It is an error to fail to provide a Column Width for all columns in the output structure, either through implicit or explicit reference.

d. Start Column Name and End Column Name refer to names assigned to columns in the output structure via these column characteristics. The span name is associated with this span of columns.

e. It is an error to specify more than one set of column characteristics for a given column, either through implicit or explicit reference.

B.4.5.9.4 Cell characteristics (cellatts). Characteristics that apply specifically to Table Cells.

Characteristics - Definitions:	Values - Definitions:
Column Name	String
Span Name	String
Span Depth	Integer

EXPLANATION:

a. Either a Column Name or Span Name should be specified. The Column Name and Span Name refer to the names assigned to a column or span of columns via the Column Characteristics. In the case where both are specified, the Column Name is used. In the case where neither is specified, the text flow rules determine the column in which the cell contents are placed.

b. The Integer value for the Span Depth Characteristic designates the number of rows the entry is to straddle.

B.4.6 COMPOSITION — GRAPHICS. This section covers the treatment of graphics. The characteristics described in this section allow a formatting system to insert graphic entities into a document according to the constraints specified by the original illustrator and the author.

B.4.6.1 Concepts.

B.4.6.1.1 Definition of terms. The Output Specification provides special characteristics for the handling of source elements used to designate illustrations or drawings, known as graphic elements. Graphic elements are composed within a special composition area on the page known as a reproduction window. Special sizing and placement characteristics are available for graphic elements. A graphic is a single illustration with no other text associated with it. A macrographic is a collection of graphics that are overlaid in a single reproduction area.

B.4.6.1.2 Sources of graphics information. A graphic element is either a non-SGML external entity (usually a graphics-encoded file) or an SGML marked-up element. Non-SGML graphic entities are usually associated with an empty element in the source DTD.

In the case of non-SGML entities, it is assumed the graphic object has an inherent “bounding box” that can be used to size and place the graphic within the reproduction window. In the case of SGML marked-up elements, such a bounding box must be defined for each block of text (element). Sizing capabilities, such as scaling, are not available for text elements. The definition of the bounding box includes its width and depth and a specification of which corner is to be used as the reference point for placement purposes. The bounding box itself is placed relative to the reproduction area. Within the bounding

box, composition characteristics are used to compose the text and are relative to the bounding box. In other words, the bounding box can be thought of as the current flowing text area.

Graphic elements often have associated information about how to display the graphic. This information can reside in several places:

a. Specified in a FOSI.
b. Specified in source document attributes.
c. Data attributes specified for the data content notation.
d. System-specific graphics parameters.

This Output Specification addresses only a and b. For any graphics characteristic, if no value is specified in the FOSI or source, it is left to the system to determine the value from other sources.

B.4.6.1.3 Interaction of the reproduction window, sizing, and placement. Specifying information for graphic element display can be thought of as a three-step process:

a. Defining a reproduction window into which the graphic element is to be placed.
b. Determining which portion of the graphic is to be displayed or determining a bounding box for text.
c. Specifying how the graphic is to be scaled and how the graphic or text block is to be positioned in the repro window.

A reproduction window can be associated with a single graphic element or with a non-graphic element that is simply a container for one or more graphic elements. In the case of a container, no sizing or placement information is specified. In the case of graphics or text blocks within the container, no repro window is specified, but sizing (for graphics) and placement information is. The graphic or text block is placed into the repro window associated with its nearest ancestor in the source document.

B.4.6.1.4 World coordinates. The world coordinate system is used to describe the two-dimensional space in which the graphic is defined and placed. The point of origin is the lower left corner of the graphic and has the coordinates (0,0). The upper right corner has the coordinates (10000,10000).

B.4.6.1.5 Assumptions. The Output Specification assumes that there is no conflict between the information provided within the graphic entity and the SGML source provided by the author.

B.4.6.1.6 Available space. The available space for placement of graphics is constrained by the FOSI Page Models. That is, graphics must be able to fit, after allowed cropping and scaling, into one of the Layout Areas specified by the FOSI.

B.4.6.2 Graphic characteristics.

B.4.6.2.1 Reproduction area dimensions. Information about the size of the reproduction area (the area on the presentation media) in which the graphic is to be placed.

Characteristics - Definitions:	Values - Definitions:
Repro Area Width	Size/Distance
Repro Area Depth	Size/Distance

EXPLANATION:

a. If the Repro area width is left #IMPLIED, the repro area width shall be considered to be the prevailing text width taking margins and left and right indents into account.
b. If the Repro area depth is left #IMPLIED, the repro area depth shall be considered whatever vertical amount is needed to contain the graphic up to a maximum of the height of the column/flowing text area.

B.4.6.2.2 Graphic sizing. Information concerning constraints on how to modify the size or view of graphics to be placed in the reproduction area.

Characteristics - Definitions:	Values - Definitions:
Graphic Name	Pointer
Horizontal Scaling	Integer
Vertical Scaling	Integer
Scale to Fit	Toggle
Lower Left Coordinates	String
Upper Right Coordinates	String

EXPLANATION:

a. The Graphic Name may be the name of an entity declared in the FOSI or it may be omitted in which case it will be filled in from the source document using a fillval construct.
b. Scaling values are interpreted as percentages. For example, a value of “80” means “80%”.
c. Scaling is disallowed by supplying a value of “0” for the Horizontal and Vertical Scaling characteristics as well as the Scale to Fit characteristic.
d. The Scale to Fit characteristic allows the graphic to be scaled as needed to fit the size of the reproduction area. Scaling is by the same factor in both directions. Horizontal and Vertical Scaling characteristics take precedence when they have values other than “0”.

e. The sizing coordinates define a section (window) of the graphic. This allows both general cropping functionality as well as the ability to designate a particular portion of the graphic to be used for an illustration.

f. The syntax to be used for the Lower Left Coordinate String is “integer,integer” where the first integer refers to the starting position of the graphic window along the horizontal axis and the second integer refers to the starting position of the graphic window along the vertical axis. Thus a Start Coordinate value of “0,0” indicates that the desired graphic window begins at the lower left point of the graphic board.

g. The syntax to be used for the Upper Right Coordinate String is “integer,integer” where the first integer refers to the ending position of the graphic window along the horizontal axis and the second integer refers to the ending position of the graphic window along the vertical axis. Thus an End Coordinate value of “10000,10000” indicates that the desired graphic window ends at the upper right point of the graphic board. A specification of “0,0” for the Lower Left and “10000,10000” for the Upper Right designates that the portion of the graphic to be used includes the entire graphic board, whereas “5000,5000” for the lower left and “10000,10000” for the upper right would signify that the upper right quadrant of the graphic would be used for the illustration.

B.4.6.2.3 Text block. Information concerning the size and reference point of a text block.

Characteristics -Definitions:	Values - Definitions:
Text Block Width	Size/Distance
Text Block Depth	Size/Distance
Horizontal reference point	List (Left, Right)
Vertical reference point	List (Top, Bottom)

EXPLANATION:

a. When Text Block Depth is not specified or set to zero, the text block has variable depth according to its content, much like a table cell.

b. The reference points specify the “fixed” corner of the text block. It is the corner placed at Coordinate Start and End in the Placement category. In the case of variable depth text blocks, the text block depth grows away from the fixed corner, that is, downward if the reference point is “top” and upward if the reference point is “bottom”.

c. Text blocks overlay the graphic.

B.4.6.2.4 Placement. Information concerning constraints on where and how to place graphics or text blocks with respect to the reproduction area.

Characteristics - Definitions:	Values - Definitions:
Horizontal Placement	List (Left, Center, Right, None)
Vertical Placement	List (Top, Middle, Bottom, None)
Start Coordinates	String
End Coordinates	String
Rotation	Toggle

EXPLANATION:

a. Horizontal and Vertical Placement are used to place graphics within a repro area. Either relative (horizontal and vertical) or specific (coordinates) positioning should be specified. That is, use of the Placement and Coordinate characteristics is mutually exclusive. When Start and End Coordinates are supplied, the values for Horizontal and Vertical Placement are ignored.

b. The Start and End Coordinate characteristics may be used for placing multiple graphics or portions of graphics in a single reproduction area. When these characteristics are used and Scale to Fit is turned “on”, the graphic is scaled to fit within the bounds of the coordinates specified instead of the entire reproduction area. If Scale to Fit is turned “off” and the graphic will not fit in the window specified by the Start and End Coordinates, then the Start Coordinate is to be used as an origin and as much of the graphics as will fit in the window is displayed. End Coordinate is ignored for text blocks, as they cannot be scaled.

c. The syntax to be used for the Start Coordinate String is “integer,integer” where the first integer refers to the starting position of the graphic along the horizontal axis and the second integer refers to the starting position of the graphic along the vertical axis. Thus a Start Coordinate value of “0,0” indicates that the lower left point of the graphic is to be placed starting at the lower left corner (point of origin) of the repro area.

d. The syntax to be used for the End Coordinate String is “integer,integer” where the first integer refers to the ending position of the graphic along the horizontal axis and the second integer refers to the ending position of the graphic along the vertical axis. Thus an End Coordinate value of “10000,10000” indicates that the graphic is to be placed such that the upper right point of the graphic coincides with the upper right corner of the repro area.

e. When the value for rotation is “off” the graphic is placed in the same orientation in which it has been received in. When the value is “on” the graphic is rotated 90 degrees counterclockwise before being placed in the repro area.

B.4.7 COMPOSITION — FOOTNOTES. The footnote description (ftndesc) section of the FOSI provides part of the specification of how footnotes will be generated and formatted.

There are several aspects to describing footnotes that must be coordinated. The footnote area element, subordinate to the flowing text area in the page description section of the FOSI, has various attributes that describe characteristics of the footnote area such as footnote placement and separators. This footnote area element has subcharacteristics to specify further the formatting of the various footnote separators. Note that the subcharacteristics in the footnote area are not used to affect the formatting of the text of the footnotes (see B.4.3.2.12.5).

The footnote description (ftndesc) section of the FOSI, contains the e-i-cs that describe the elements (or pseudo-elements) that will cause their contents to be placed in the footnote area of the page on which these elements are used. Associated with each e-i-c in the ftndesc is a ftnatts element which contains a charlist (minus keeps and span) that specifies the formatting of the footnote content itself. More precisely, the contents (for example, the charlist) of the e-i-c in the ftndesc determines the characteristics of what gets placed in the flowing text area when this e-i-c instance is encountered; whereas the contents (the charlist) of the ftnatts associated with this e-i-c determines the characteristics for what gets placed in the footnote area when this e-i-c is encountered. The content of this e-i-c instance (that is, the source document element's content) is placed into the footnote area under control of the characteristics defined by the ftnatts (unless the ftnatts' charlist uses the Suppress characteristic).

There are two basic tagging schemes in general use for footnotes. In the simpler case where the source DTD has defined an element whose content is the footnote text and whose location in the document instance determines where the footnote callout is to be placed, this element would be described in the ftndesc and no other element is required. In the more complex situation such as that used in the Example DTD in MIL-HDBK-28001, there is one element (`ftnote`) whose content is the footnote text but whose position in the document instance is irrelevant and another element (`ftnref`) with no content but whose position determines both the location of the callout in the flowing text and the page in whose footnote area the footnote text will be placed. The `ftnref` element has an IDREF attribute that refers to the `ftnote` element's ID attribute to allow the `ftnref` to reference the appropriate footnote text. To accomplish this, the `ftnote` e-i-c must save its contents for later use by the `ftnref` e-i-c. The `ftnref` element must both cause the footnote callout number to appear in the flowing text and cause the previously saved footnote text to be placed in the footnote area of the current page. Neither the `ftnote` e-i-c nor the `ftnref` e-i-c are special in that they both appear in the styldesc part of the FOSI (and the e-i-c for a pseudo-element whose use is triggered by the `ftnref` element appears in the ftndesc). See MIL-HDBK-28001 for examples showing footnote descriptions for both styles of footnote tagging.

B.4.8 COMPOSITION — LOOSELEAF CHANGE PACKAGES. This section describes the characteristics necessary for the composition of looseleaf change packages. Looseleaf change packages are treated separately in this specification because they have unique formatting and processing characteristics. Looseleaf change packages for technical documents provide an economical mechanism to distribute changes to technical documents. Production and distribution of change packages consisting of only the changed pages costs less than

production and distribution of entire technical documents. Most military technical documents are distributed and maintained as looseleaf documents.

To facilitate their updating, looseleaf technical documents are not bound like normal books. A looseleaf technical document is distributed as an unbound collection of pages with holes drilled in them, which allows them to be held in ring-binders. Looseleaf storage of technical documents allows them to be easily updated as it is a relatively simple operation to add and remove pages.

Composition of a looseleaf document differs from composition of normal documents. Original page boundaries must be honored as a document goes through the composition process. Added material that overflows a page must be placed on a new page, where this new page's folio must indicate that it is an overflow page. Each changed page must be annotated with its current change level, and text and illustration changes may be highlighted. In addition, filing instructions or a list of effective pages (LEP) must be generated to serve as a set of user instructions for inserting the changed pages into the technical document.

A typical change package for a technical document consists of an updated cover or title page, a promulgation letter, a list of effective pages or a set of filing instructions, an updated table of contents, an updated list of illustrations, an updated list of tables, and the actual changed pages with their appropriate markings. If the technical document is printed two-sided (that is, on both sides of a piece of paper), backing pages must be provided where necessary so that complete leaves are distributed.

The Change Description section provides categories that connect change information in the instance to the composition system, so that change levels can be tracked during composition and retrieved for formatting. This section also provides categories that describe how to generate data for LEPs, how to link source instance tags indicating page boundaries to the composition system, and how to construct page folios.

B.4.8.1 Tagging source instance changes. In order for the composition system to be able to produce change packages, the source instance must be annotated with change information. This change information falls into these categories:

a. Current change level. The current change level for the technical manual must be specified. This change level is usually represented as a non-negative integer, but may be an arbitrary string.
b. Change level history. In order for a composition system to determine the highest change level on a page, an ordering of change levels must be established. If change levels are integers, the ordering exists. However, if change levels are represented as names (strings), the change level name ordering must be explicitly established.
c. Tagging of text changes. Any text characters to be changed must be annotated with appropriate change tagging, so that the composition system can detect the change. Text changes must also be associated with a specific change level.
d. Tagging of container changes. Some elements such as figures and aggregates such as sections, whose primary content is not characters, must be extended to

support an attribute indicating that the container element has changed. Marking of container elements is done when a container element is inserted or deleted.

e. Indicating page boundaries. Boundaries between pages must be marked with a special tag, usually referred to as the page break tag. This page break tag must include sufficient information for the composition system to reestablish folio and change information.

The elements and attributes in a source instance that represent information of the categories listed must be identified to the composition system. The change section in the FOSI performs this identification function.

B.4.8.2 The structure of the change section. The Change Description (chngdesc) section of the FOSI contains the categories necessary to produce looseleaf change packages. There are four parts to chngdesc:

a. Chnglevl establishes associations between FOSI counters and string variables used to access change information and the elements and attributes in the source instance that indicate changes.

b. Lepdesc specifies how LEP information is gathered and formatted as a string containing pseudo-elements and CDATA. Formatting of a LEP for composition is under control of the FOSI e-i-cs for these pseudo-elements.

c. Pgbrk specifies the association of page break information in the source instance and its processing by the composition processor

d. Chnginst specifies the components of added page folios and associates page break information in the source instance with folio components

B.4.8.2.1 Change level. The Chnglevl category specifies the elements and attributes in the source instance used to specify change information and the FOSI counters and strings used to retrieve change information for a portion of the technical document.

Change levels can be expressed as numbers (for example, non-negative integers such as “0” or “5”) or as names (for example, arbitrary strings such as “E” or “June 14, 1994”). Every change name will always be implicitly associated with a change number and the mapping will be one-to-one. This association is needed to provide the ordering of change strings necessary to determine the highest change level of all changes occurring in a portion of the technical document (for example, page).

Access to the current change level for a portion of (or the entire) technical document is made by referencing a FOSI variable identified in the Chnglevl category. Different variables are used for access to change level numbers and change level names. Thus, a FOSI has access to both change level numbers and names. These FOSI variables are read-only.

Chnglevl has three subcategories:

a. Lvlnames specifies the FOSI variables that will be used to access change level information

b. Chngtag identifies a tag and its attributes that denote a change (for example, tag "change")

c. Chngatt identifies an attribute that denotes a change to its associated element (for example, attribute "inschlvl")

Characteristics - Definitions:	Values - Definitions:
Document Change Level Tag - Tag containing document's change level	Name
Document Change Level Attribute - Attribute of Document Change Level Tag containing document's change level	Name
Document Change Level Name Access String Variable - Name of FOSI string variable used to access document change level name	Name
Page Change Level Name Access String Variable - Name of FOSI string variable used to access page's change level name	Name
Leaf Change Level Name Access String Variable - Name of FOSI string variable used to access leaf's change level name	Name
Spread Change Level Name Access String Variable- Name of FOSI string variable used to access spread's change level name	Name
Document Change Level Number Access Variable - Name of FOSI counter or string variable used to access document change level number	Name
Page Change Level Number Access Variable - Name of FOSI counter or string variable used to access page's change level number	Name
Leaf Change Level Number Access Variable - Name of FOSI counter or string variable used to access leaf's change level number	Name
Spread Change Level Number Access Variable - Name of FOSI counter or string variable used to access spread's change level number	Name

Characteristics - Definitions:	Values - Definitions:
Format Change Level Name Delimiter - Name of FOSI pseudo-element that implicitly delimits references to FOSI change level name access variables	Name
Format Change Level Number Delimiter - Name of FOSI pseudo-element that implicitly delimits references to FOSI change level number access counters and string variables.	Name

EXPLANATION:

a. The current change level of the technical document must be made available to the formatting system. This change level information can be presented as either the content of a source instance element or the value of an attribute of a source instance element. The Document Change Level tag identifies the source level element. If only a Document Change Level Tag is specified, then the content of this tag is used as the current change level. If both a Document Change Level Tag and a Document Change Level Attribute are specified, then the value of the attribute of the tag is used as the current change level.

b. The current change level of the technical document can be specified as either a number or a name. The composition processor uses a list of valid change level names in the source instance to determine whether the specified value is to be treated as a number or a name. If the current change level value appears in this list, it is assumed to be a name and its corresponding change level number is inferred. Otherwise, the document's current change level is treated as a number.

c. When the FOSI needs to access the current change level name for the entire document, the current page, the current spread, or the current leaf, the appropriate FOSI string variable should be referenced. References to the Document Change Level Name Access String Variable yields the document's current change level name. Similarly, references to the Page Change Level Name String Variable yields the current page's change level name, references to the Leaf Change Level Name Access String Variable yields the current leaf's change level name, and references to the Spread Change Level Name Access String Variable yields the current spread's change level name. Read-only access is provided to these string variables.

d. When the FOSI needs to access the current change level number for the entire document, the current page, the current spread, or the current leaf, the appropriate FOSI string counter or string variable should be referenced. References to the Document Change Level Number Access Variable yields the document's current change level number. Similarly, references to the Page Change Level Number Access Variable yields the current page's change level number, references to the Leaf Change Level Number Access Variable yields the current leaf's change level

number, and references to the Spread Change Level Number Access Variable yields the current spread's change level number. Read-only access is provided to these counters and string variables.

e. In order to simplify FOSI formatting of change level information, change level access counters and string variables can be implicitly associated with pseudo-elements. This facility allows for pseudo-element e-i-cs to add text to the change information. For example, a pseudo-element e-i-c can add the literal text "Change " in front of a change level number. The Format Change Level Name Delimiter specifies the name of the pseudo-element that implicitly surrounds references to change level name access variables. The Format Change Level Number Delimiter specifies the name of the pseudo-element that implicitly surrounds references to change level number access variables.

f. In order to handle the situation where no change information is formatted for original (change 0) change levels, the Format Change Level Name Delimiter and Format Change Level Number Delimiter are only invoked for non-original change levels. That is, change levels other than original or change 0.

B.4.8.2.1.1 Level names. The Level Names category specifies the portion of the source instance that contains the list of change level names. If only change level numbers are used, this category should be omitted.

Characteristics - Definitions:	Values - Definitions:
List of Level Names Container Tag - Tag containing children tags that hold the individual change level names.	Name
Sub Container Tag - Name of optional tag that contains an Individual Level Name Tag.	Name
Individual Level Name Tag - Name of tag that contains a single change level name	Name
Individual Level Name Attribute - Attribute name of Individual Level Name Tag that specifies a single change level name	Name
Change Level Number Tag - Name of tag that contains the change level number for the corresponding change level name.	Name

Characteristics - Definitions:	Values - Definitions:
Change Level Number Attribute - Name of attribute that contains the change level number for the corresponding change level name.	Name
Support Change Level Name for Level 0 - Toggle indicating whether a change level name to number association exists for change 0.	Toggle

EXPLANATION:

a. The List of Level Names Container Tag characteristic contains the name of the source instance element that is the container element for the list of all change level names. Child elements, named by the contents of the Individual Level Name Tag characteristic, contain a single change level name.

```
<!-- FOSI -->
<lvlnames cntnrtag="changehistory" nametag="changelevelname">
<!-- Source Instance -->
<changehistory>
<changelevelname>20 August 1990
<changelevelname>10 January 1992
<changelevelname>15 March 1995
```

b. Change level names must appear in ascending change level order. That is, the current change appears last. Explicit mappings of change level names to change level numbers can be established by specification of Change Level Number Tag or Change Level Number Attribute.

c. A change level name can be specified by an attribute on the Individual Level Name Tag. In this case, the Individual Level Name Attribute specifies the name of the attribute.

```
<!-- FOSI -->

<lvlnames cntnrtag="changehistory" nametag="changeinfo"
 nameattr="levelname">
<!-- Source Instance -->
<changehistory>
<changeinfo levelname="20 August 1990">
<changeinfo levelname="10 January 1992">
<changeinfo levelname="15 March 1995">
```

d. In situations where child elements of the List of Level Names Container Tag do not contain change level names, but grandchild elements do, the child's tag name is specified by the Sub-Container Tag.

```
<!-- FOSI -->
<lvlnames cntnrtag="changehistory" subcntnr="changeinfo"
 nametag="levelname">
```

```
<!-- Source Instance -->
<changehistory>
<changeinfo><levelname>20 August 1990
<changeinfo><levelname>10 January 1992
<changeinfo><levelname>15 March 1995
```

e. The value of an attribute of the grandchild element can specify a change level name via the Individual Level Name Attribute.

```
<!-- FOSI -->
<lvlnames cntnrtag="changehistory" subcntnr="changeinfo"
 nametag="level" nameattr="levelname">
<!-- Source Instance -->
<changehistory>
<changeinfo><level levelname="20 August 1990">
<changeinfo><level levelname="10 January 1992">
<changeinfo><level levelname="15 March 1995">
```

f. By default individual change level names must appear in ascending change level order and are associated with change level numbers starting with 1. Change level names however can be associated with explicit change level numbers. Explicit mappings of change level names to change level numbers provide for change level number ranges other than the consecutive integers one, two, three, and so on. Also, an explicit mapping need not be presented in ascending change level order.

g. Explicit mapping of change level name to change level number can be specified either by tag content or attribute value.

h. To explicitly map change level names to change level numbers, with the change level numbers appearing as the content of tags, the tag's name is specified as the value of the Change Level Number Tag characteristic.

```
<!-- FOSI -->
<lvlnames cntnrtag="changehistory" subcntnr="changeinfo"
 nametag="levelname" numbtag="levelnumber">
<!-- Source Instance -->
<changehistory>
<changeinfo><levelname>20 August 1990<levelnumber>1
<changeinfo><levelname>10 January 1992<levelnumber>2
<changeinfo><levelname>15 March 1995<levelnumber>3
```

i. To explicitly map change level names to change level numbers, with the change level numbers appearing as the value of attributes, the attribute's name is specified as the value of the Change Level Number Attribute characteristic.

```
<!-- FOSI -->
<lvlnames cntnrtag="changehistory" subcntnr="changeinfo"
nametag="level"
 nameattr="levelname" numbattr="levelnumber">
<!-- Source Instance -->
<changehistory>
<changeinfo><level levelname="20 August 1990" levelnumber="1">
```

```
<changeinfo><level levelname="10 January 1992" levelnumber="2">
<changeinfo><level levelname="15 March 1995" levelnumber="3">
```

j. Explicit mappings of change level names to change level numbers can be specified with one specified as content and the other specified as an attribute value.

k. The Support Change Level Name for Level 0 toggle determines whether or not a change level name is associated with change 0. If this toggle is one and change level names are implicitly mapped to change level numbers, the first change level name is associated with change level zero. If this toggle is zero and change level names are implicitly mapped to change level numbers, the first change level name is associated with change level one.

B.4.8.2.1.2 Change tags. The Change Tags category specifies a source instance element whose character content has changed.

Characteristics - Definitions:	Values - Definitions:
Change Tag List - List of tag names used to surround changed content.	Name List
Change Level Type - The type of change level specified by the tags in Tag List.	List (Name, Number)
Change Level Attribute - The name of the attribute used to specify the change level on tags in the Tag List.	Name
Change Type - The type of change specified by the tags in Tag List.	List (Insert, Delete, Change)
Change Type Attribute Name - The name of the attribute (on tags in the Change Tag List) used to specify the type of change.	Name

EXPLANATION:

a. The names of tags used to surround changed data content appear in the Change Tag List.

b. Change level information must be specified with each change tag and the type of change level is specified by the Change Level Type characteristic. The Change Level Type is either change level name or change level number.

c. The change level name or number must be specified as the value of an attribute of each change tag. The name of this attribute is specified as the value of the Change Level Attribute characteristic.

d. Character data may be inserted, deleted, or changed (both inserted and deleted). A single change tag can be used to indicate any of the three types of changes or different change tags can be used to distinguish among the types of changes.

e. To use a single tag for all types of changes, the type of change must be specified as the value of an attribute on the change tag. This attribute's name is given as the value of the Change Type Attribute Name. (The value of this attribute must be one of the values given for Change Type.)

f. To associate tags in the Change Tag List with a single type of change, specify the type of change in the Change Type Attribute characteristic.

g. A sample Change Tag category for a standard change tag is as follows. The tag name is change; the change level is specified by the level attribute, the change level is specified by name, and the type of change is specified by the change attribute.

```
<!-- Source instance usage -->
<para>Some text <change level="E" change="delete">XXX</change>
<!-- FOSI -->
<chngtag levltype="name" taglist="change" levlattr="level"
 chngtype="change">
```

B.4.8.2.1.3 Change attributes. The Change Attributes category specifies source instance elements whose content is considered changed when an attribute on those elements is specified. Change Attributes are used for container elements, whose content is not character data.

Characteristics - Definitions:	Values - Definitions:
Tag List - List of tag names for which changes are indicated by providing an attribute value.	Name List
Attribute List - List of attribute names which indicate a change to its element when the attribute is given a value.	Name List
Change Type - The type of change specified by the tags in Tag List.	List (Insert, Delete, Change)
Change Level Source - The source of the change level information.	List (Level, Switch)
Change Level Type - The type of change level specified by the attributes in Attribute List. Only applicable if Change Level Source is Switch.	List (Name, Number)

EXPLANATION:

a. The names in Tag List identify a collection of tags whose change status is indicated by the presence of any attribute names in Attribute List.

b. The type of change is indicated by Change Type.

c. The change level for the tag can either be defaulted from the document's change level (Level) or provided by the value of the attribute from Attribute List (Switch). If the change level is provided by the value of an attribute, the type of change level must be specified in Change Level Type.

d. Omission of the Tag List characteristic implies that the change attributes apply to all elements.

e. A sample Change Attributes category for the standard inschlvl and delchlvl attributes:

```
<chngatt attrlist="inschlvl" chngtype="insert">
<chngatt attrlist="delchlvl" chngtype="delete">
```

B.4.8.2.1.3.1 Changed elements characteristics. When an element is specified as changed, a set of characteristics and ATTs can be conditionally merged. This conditional merging can be done for elements listed in Change List category and Change Attribute category.

Characteristics - Definitions:	Values - Definitions:
When Triggered - When to trigger the merging of characteristics and evaluation of ATTs based on the comparison of the document's change level to the changed element's change level.	List (Less, Equal, Greater)

EXPLANATION:

a. Use of Changed Elements Characteristics can simplify the specification of e-i-cs. The Changed Elements Characteristics category provides a way to specify in one place a collection of characteristics and ATTs that will apply to a group of e-i-cs.

b. These characteristics are applied only if the When Triggered condition is satisfied. If the When Triggered condition is satisfied the characteristics are applied after all other characteristic merging, including inheritance and defaulting.

B.4.8.2.2 List of effective page description. The List of Effective Page Description (Lepdesc) category describes how to gather and organize page information for presentation in a LEP. This LEP data is created on demand and made available as a pseudo-element delimited string, accessible via a time-independent FOSI string variable. A Lepdesc does not actually control the formatting of a LEP. LEP formatting is specified by creating Style Description e-i-cs for the pseudo-elements appearing in the LEP data string.

Although some LEPs may contain only one LEP entry per document page, a more common situation is for similar pages to be grouped together in a single LEP entry. In order to support a variety of page groupings, a number of Lepdesc characteristics provide control information for page grouping. The basic rules for page grouping in a LEP entry are:

a. Grouping of pages is enabled.

b. The pages' folios share a common prefix.

c. The pages' change levels are the same.

d. None of the pages is blank (subject to the inclusion of blank pages as specified by the Group Blank Pages characteristic).

Multiple Lepdescs can be present in a FOSI. In order to simplify the creation of formatting e-i-cs for Lepdesc generated pseudo-elements, these pseudo-elements' names are declared within the Lepdesc.

Characteristics - Definitions:	Values - Definitions:
String Variable Name - Name of FOSI string variable used to access LEP data	Name
Scope - Name of source instance element whose content determines which pages are to represented in LEP.	Name
List of Page Sets - Generate LEP page entries only for pages from page sets listed in this characteristic.	Names
LEP Pseudo-Element Name - Name of pseudo-element that delimits the generated LEP data.	Names
LEP Entry Pseudo - Element Name - Name of pseudo-element that delimits one LEP entry.	Name
Group Pages Flag - Determines how pages are grouped into a single LEP entry.	List (No, Level, Status)
Page Prefix String Variable - Name of FOSI string variable that contains page folio prefix used to determine grouping of pages.	Name
Blank Page Flag - Toggle indicating whether or not blank pages will appear in LEP.	Toggle
Deleted Page Flag - Toggle indicating whether or not deleted pages will appear in LEP.	Toggle
Group Blank Pages - Determines the conditions under which blank pages participate in page grouping.	List (No, Yes, End, Begin, Endbegin)

Characteristics - Definitions:	Values - Definitions:
Leaf Level Flag - Toggle that determines whether an individual page's change level or that page's leaf's change level is used for grouping pages.	Toggle
First Folio String Variable Name - Name of FOSI string variable used to access first folio of current LEP entry during LEP Entry Pseudo-Element Name's e-i-c processing.	Name
Last Folio String Variable Name - Name of FOSI string variable used to access last folio of current LEP entry during LEP Entry Pseudo Element Name's e-i-c processing.	Name
Change Level Number Variable - Name of FOSI counter or string variable used to access change level number of this LEP entry's page(s).	Name
Change Level Name String Variable - Name of FOSI string variable used to access change level name of this LEP entry's page(s).	Name
Page Status String Variable - Name of FOSI string variable used to determine status of this LEP entry's page(s).	Name
Single Page Flag Variable - Name of FOSI string variable used to determine status as to whether there is one or more than one page represented by this LEP entry.	Name

EXPLANATION:

a. Each Lepdesc must specify a FOSI string variable name via which the generated LEP data can be accessed. If more than one Lepdesc is specified, each must have its own FOSI string variable.

b. Scope is used to restrict which pages are included in the LEP. If Scope is specified only pages that contain content from the named element are included in the LEP.

c. By default, all pages in the document are included in the LEP. The pages included in the LEP can be restricted to a group of page sets by listing those page set names in the List of Page Sets.

d. LEP Psuedo-Element Name and LEP Entry Pseudo-Element Name specify the names of the two pseudo-elements used to delimit LEP processing. For example, if

the LEP Pseudo-Element Name is set to “lepa” and the LEP Entry Pseudo-Element Name is set to “lepaentry”, then the string created would have the following form: `"<lepa><lepaentry></lepaentry>...<lepaentry></lepaentry></lepa>"`

e. The Group Pages Flag determines how pages are grouped into a single LEP entry. If this flag is set to “No”, no grouping of pages is done. If this flag is set to “Level”, pages are grouped by change level. (Most LEPs have their pages grouped by change level.) If this flag is set to “Status”, pages are grouped by page status.

f. The Page Prefix String Variable names the FOSI string variable that contains the page folio’s prefix (seeB.4.8.2.4).

g. The Blank Page Flag determines whether or not blank pages will appear in the LEP. If this flag is set to one (“1”), blank pages will appear in the LEP. If this flag is set to zero (“0”), blank pages will not appear in the LEP.

h. Deleted Page Flag determines whether or not deleted pages will appear in the LEP. If this flag is set to one (“1”), deleted pages will appear in the LEP. If this flag is set to zero (“0”), deleted pages will not appear in the LEP.

i. The Group Blank Pages characteristic determines whether or not blank pages are grouped into LEP entries, and if blank pages are grouped in LEP entries, how they are grouped. When “NO” is specified, blank pages will be placed in a separate LEP entry. When “YES” is specified, the "blankness" of a page is not considered in grouping. When “END” is specified, a group may end with a single blank page. When “BEGIN” is specified, a group may start with a single blank page. When “ENDBEGIN” is specified, a group may begin or end with a single blank page.

j. The Leaf Level Flag determines how the two pages of a leaf participate in page grouping, when the two pages are at different change levels. When this flag is “0” (zero, false), each page’s change level will be used to determine grouping. When this flag is “1” (one, true), each page of a leaf will be considered to be at the highest change level of that leaf for grouping purposes.

k. Generation and formatting of a LEP occurs when a FOSI e-i-c does a usetext reference to the FOSI string variable named in the String Variable Name characteristic. The usetext reference to this string variable causes the FOSI processor to gather the page information and create the string containing the appropriate pseudo-element names. For every page grouping there will be one occurrence of the pseudo-element named in the LEP Entry Pseudo-Element Name characteristic. Usetext processing of this string will cause this pseudo-element’s e-i-c to be triggered once for every page grouping. During e-i-c processing for each LEP Entry Pseudo-Element, FOSI counters and string variables will be bound to values representing information for the current LEP Entry’s page grouping. The e-i-c can thus reference these counters and string variables to construct a correctly formatted LEP entry.

l. The First Folio String Variable Name specifies the name of the string variable containing the page folio for the first page of this LEP entry’s page range.

m. The Last Folio String Variable Name specifies the name of the string variable containing the page folio for the last page of this LEP entry's page range. If there is only one page in the page range, the first and last page folios are the same folio.

n. The change level for the current LEP entry's page range is made available in both change level number and change level name form. The change level number is contained in the FOSI counter or string variable specified in the Change Level Number Variable characteristic. The change level name is contained in the FOSI string variable specified in the Change Level Name String Variable characteristic.

o. The status of the current LEP entry's page range is contained in the FOSI string variable specified in the Page Status String Variable characteristic. The status is one of the following strings: UNCHANGED, CHANGED, ADDED, or DELETED.

p. The FOSI string variable specified in the Single Page Flag Variable is set to zero if there is more than one page in this LEP entry's page range and one if there is only one page in this LEP entry.

B.4.8.2.3 Page break element description. The Page Break Element Description controls the form and content of a page break tag which marks the page boundaries in the source instance. Page break tags may appear between tags or within character data. Page break tags are used by the composition application to re-create identical page breaks on re-composing the document.

Characteristics - Definitions:	Values - Definitions:
Page Break Tag Name - Name of source instance tag used to represent page breaks.	Name
Page Break Tag Generation - Determines where page break tags should be inserted.	List (Recto, Rectoverso, Verso)

EXPLANATION:

a. The name of the page break tag used in the source instance is specified in the Page Break Tag Name characteristic.

b. Some composition applications may create an augmented copy of the source instance. This augmented copy might include page break tags. The Page Break Tag Generation characteristic determines where page break tags should be placed in this augmented copy. There are three choices: recto - at the top of recto pages only, rectoverso - at the top of all pages, and verso - at the top of verso pages only. During a later composition of this augmented copy these page break tags allow text to flow within a leaf (recto), a page (rectoverso), and a spread (verso).

B.4.8.2.3.1 Page break attribute description. The Page Break Attribute Description establishes an association between a page break tag attribute and a specific piece of information needed by the composition application to perform looseleaf composition.

APPENDIX B

Characteristics - Definitions:	Values - Definitions:
Attribute Name - Name of attribute appearing on source instance page break tag.	Name
Attribute Type - Type of attribute	List (Folio, Foliodesc, Nextfolio, Nextfoliodesc, Sequence, Prefix, Parabreak, Hyphen, Shrink, Rectoverso, Change, Changename, Security, Empty, Ignore)

EXPLANATION:

a. A page break tag attribute, whose value is needed for looseleaf composition, has its name specified as the value of the Attribute Name characteristic. There are a fixed set of composition attribute types and each page break tag attribute must be associated with one of those types via Attribute Type characteristic specification.

b. The folio attribute type value is a string to be used as a page folio.

c. The foliodesc attribute type value is a folio description.

d. The nextfolio attribute type value is the next page's folio. The next page's folio is needed for generation of folios such as "5-1/(5-2 blank)" .

e. The sequence attribute type value is the page sequence number within the pages sharing the same page prefix.

f. The prefix attribute type value is the page folio prefix string.

g. The parabreak attribute type value is a toggle indicating whether or not the last paragraph on this page breaks over to the next page. A value of zero ("0") indicates that the last paragraph does not break and a value of one ("1") indicates that the last paragraph does break.

h. The hyphen attribute type value is a toggle indicating whether or not the last word on the last line of the last page is hyphenated and the rest of the word appears on the next page. A value of zero ("0")indicates that the last word is not hyphenated and a value of one ("1") indicates that the last word is hyphenated.

i. The shrink attribute type value is a shrink factor expressed as an integer percentage. A shrink factor can be used to shrink vertical space in the page body so as to pull back text on this page that has previously overflowed onto the next page.

j. The rectoverso attribute type value is a toggle indicating whether or not this page is a recto or verso page. A value of zero ("0") indicates a verso page and a value of one ("1") indicates a recto page.

k. The change attribute type value represents the page's change level.

l. The security attribute type value represents the page's security level.

m. The empty attribute type value is a toggle indicating whether or not this page is empty (blank). A value of zero ("0") indicates a non-empty (non-blank) page and a value of one ("1") indicates an empty (blank) page.

n. The ignore attribute type value is a toggle indicating whether or not this page break tag should be ignored during composition for the purposes of generating overflow pages. A value of zero (“0”) indicates the page break tag should be treated as a page boundary and overflow pages generated as needed. A value of one (“1”) indicates that this page break tag should be ignored.

B.4.8.2.4 Change instruction description. The Change Instruction Description specifies the components of added page folios and associates page break information in the source instance with folio components needed for looseleaf composition.

Characteristics - Definitions:	Values - Definitions:
Folio String Variable Name - Name of FOSI string variable used to access the current page’s complete folio string.	Name
Next Folio String Variable Name - Name of FOSI string variable used to access the next page’s complete folio string.	Name
Folio Prefix String Variable Name - Name of FOSI string variable that contains the page folio’s prefix.	Name
Page Sequence Number - Name of FOSI counter that contains the page’s sequence number within a page group established by page folio prefix.	Name
Folio String Constructor Name - name of FOSI pseudo-element used to construct the folio string for this page.	Name
Overflow Page Counter Names - Ordered list of FOSI counter names that provide overflow page counts.	Names
Overflow Page Folio Pseudo-Element Names - Ordered list of pseudo-element names used to construct overflow page folio strings. First level overflow page folio strings are constructed by the first entry in this list; second level overflow page folio strings are constructed by the second entry in this list; and so on.	Name
Insert Overflow Leaf Flag - Determines whether an overflow page of leaf is generated.	List (Yes, No, Perpgdesc)

APPENDIX B

EXPLANATION:

a. The Folio String Variable Name specifies the FOSI string variable used to access the current page's complete folio string. In order to support military specification blank page folio styles, the next page's complete folio string can be accessed by the FOSI string named in the Next Folio String Variable Name.

b. The Folio Prefix String Variable Name specifies the FOSI string variable used to access the current page folio's prefix. This string is needed for grouping of pages for LEP processing as well as construction of page folio strings.

c. The Folio String Constructor Name specifies the name of the FOSI pseudo-element that is used to construct the folio string for this page. A corresponding e-i-c must be provided in the style description section.

d. The List of Overflow Page Counter Names specifies an ordered list of FOSI counters used to track overflow page sequence numbers. First level overflow page sequence numbers are tracked in the first named counter; second level overflow sequence numbers are tracked in the second named counter; and so on.

e. The List of Overflow Page Folio Pseudo-Element Names specifies a list of FOSI pseudo-element names used to construct the entire folio string for an overflow page. Corresponding e-i-c's must be provided in the style description section. First level overflow page folio strings are constructed by the first pseudo-element name's e-i-c; second level overflow page folio strings are constructed by the second pseudo-element's e-i-c; and so on.

f. The Insert Overflow Leaf Flag determines whether overflow pages or leaves are generated. If this flag is Yes, overflow leaves will be generated. Generation of overflow leaves always yields an even number of pages, and if necessary the last leaf will be completed with a blank verso page. If the flag is "No", overflow pages are generated without regard to generation of leaves; thus, either an odd or even number of pages can be generated.

B.5 OUTPUT SPECIFICATION DTD

B.5.1 THE OS DTD.

```
<!DOCTYPE outspec [

<!-- The following set of declarations may be referred to using a
public entity as follows:

<!DOCTYPE outspec PUBLIC
   "-//USA-DOD//DTD OUTPUT SPEC MIL-PRF-28001 REV C 19960815//EN">
-->

<!-- NOTE:  In order to parse the following Document Type Declaration
Subset alone, insert:

         <!DOCTYPE outspec [

at the beginning of the file and append the associated "]" to the end
of the file.  -->

<!-- The following public character entity sets can be used to
specify special characters in characteristic values of type "string".-->

<!ENTITY % ISOlat1 PUBLIC "ISO 8879:1986//ENTITIES Added Latin 1//EN">

<!ENTITY % ISOpub PUBLIC "ISO 8879:1986//ENTITIES Publishing//EN">

<!ENTITY % ISOgrk3 PUBLIC "ISO 8879:1986//ENTITIES Greek Symbols//EN">

<!ENTITY % ISOnum PUBLIC "ISO 8879:1986//ENTITIES Numeric and Special Graphic//EN">

<!ENTITY % ISOtech PUBLIC "ISO 8879:1986//ENTITIES General Technical//EN">

%ISOlat1; %ISOpub; %ISOgrk3; %ISOnum; %ISOtech;

<!ENTITY % yesorno "NUMBER">

<!-- An output spec is made up of sections describing layout
characteristics for page models, and style characteristics for
graphics, tables, and all other elements. Fosicite is
optionally available to identify the FOSI -->

<!ELEMENT outspec   - o  ((rsrcdesc?, secdesc*, pagedesc?, styldesc, tabdesc?,
                          grphdesc?, ftndesc?, chngdesc?), ospackage*)>

<!ATTLIST outspec
        fosicite       CDATA -- a string --          #IMPLIED>
```

```
<!ELEMENT ospackage - o  (rsrcdesc?, secdesc*, pagedesc?, styldesc?,
                          tabdesc?, grphdesc?, ftndesc?, chngdesc?)
                                                  -(hyphrule | floatloc)>

<!-- On ospackage, the pkgname attribute is optionally available to identify
the ospackage -->

<!ATTLIST ospackage
        pkgname     ID -- a label/name --         #IMPLIED>

<!-- This resource description section gives document-wide hyphenation
rules as well as descriptions of character fills and counters that will
be used throughout the FOSI. -->

<!ELEMENT rsrcdesc  - o (hyphrule*, charfill*, counter*, stringdecl*, floatloc*)>

<!ELEMENT hyphrule  - o   EMPTY>

<!ATTLIST hyphrule
        language    ID -- an id --                #REQUIRED
        hjdata      ENTITY -- a pointer --        #IMPLIED
        wordbrk     ENTITY -- a pointer --        #IMPLIED
        unbrkwrd    ENTITY -- a pointer --        #IMPLIED
        brkchars    CDATA -- a string --          #IMPLIED
        brkbfchr    CDATA -- a string --          #IMPLIED
        brkafchr    CDATA -- a string --          #IMPLIED
        nobrkchr    CDATA -- a string --          #IMPLIED
        type        (dict |  logic |  both | any) #IMPLIED
        zone        CDATA -- a size --            #IMPLIED
        ladder      NUMBER -- an integer --       #IMPLIED
        minleft     NUMBER -- an integer --       #IMPLIED
        minpush     NUMBER -- an integer --       #IMPLIED
        clbrkok     %yesorno -- toggle --         #IMPLIED
        pgbrkok     %yesorno -- toggle --         #IMPLIED>

<!ELEMENT charfill  - o  EMPTY>

<!ATTLIST charfill
        literal     CDATA -- a string --          #IMPLIED
        orient      (vert |  horiz)               #IMPLIED
        type        (rr |  rf |  ff |  fr)        #IMPLIED
        spbefore    CDATA -- a size --            #IMPLIED
        spafter     CDATA -- a size --            #IMPLIED
        padding     CDATA -- a size --            #IMPLIED
        truncat     %yesorno -- a toggle --       #IMPLIED
```

```
        suppress    NUMBER -- integer --          #IMPLIED
        align       %yesorno -- toggle --         #IMPLIED
        cfid        NMTOKEN -- an id --           #REQUIRED
        mincount    NUMBER -- an integer --       #IMPLIED
        break       (none | before | within | after |
                     befin | inaft | befaft | anywher )     #IMPLIED>

<!ELEMENT counter   - o  EMPTY>

<!ATTLIST counter
        initial     NUMBER -- an integer --       #IMPLIED
        style       (arabic | romanuc | romanlc | alphauc |
                     alphalc | userdef)  #IMPLIED
        specstyl    CDATA -- a string --          #IMPLIED
        seq         (1 | 2)                       #IMPLIED
        except      CDATA -- a string --          #IMPLIED
        enumid      NMTOKEN -- an id --           #REQUIRED
        padlen      NUMBER -- an integer --       #IMPLIED>

<!ELEMENT stringdecl - o  EMPTY>

<!ATTLIST stringdecl
        textid      NMTOKEN  -- an id --     #REQUIRED
        literal     CDATA -- a string --     #IMPLIED
        status      %yesorno -- a toggle --  #IMPLIED
        scope       NAMES -- a list of element names -- #IMPLIED
        export      %yesorno -- a toggle --  #IMPLIED>

<!ELEMENT floatloc  - o  EMPTY>

<!ATTLIST floatloc
        floatid     ID  -- id of this location -- #REQUIRED
        floattyp    (once | atpgbrk | atcolbrk | multiref) #IMPLIED
        maxdepth    CDATA   -- a dimension --     #IMPLIED
        minspace    CDATA   -- a dimension --     #IMPLIED
        nomspace    CDATA   -- a dimension --     #IMPLIED
        maxspace    CDATA   -- a dimension --     #IMPLIED>

<!-- This section describes the priority ordering of security
values, which are placed in header and footer areas. -->

<!ELEMENT secdesc - o (secatt | sectoken)+>

<!ATTLIST secdesc
        attspec     NAMES              #IMPLIED
        secorder    NMTOKENS           #IMPLIED
```

```
        ignoresec   CDATA               #IMPLIED
        secid       ID                  #IMPLIED>

<!ELEMENT sectoken  - o EMPTY>

<!ATTLIST sectoken
        secval      NMTOKEN             #REQUIRED
        sectext     CDATA               #IMPLIED>

<!ELEMENT secatt - - (markval*, markfill?, markover?, marktext?)>

<!ATTLIST secatt    logic  (and|or)     'and'>

<!ELEMENT markval - o EMPTY>

<!ATTLIST markval
        scope       (document| sheet | page)   "page"
        attval      CDATA               #IMPLIED>

<!ELEMENT markfill - o EMPTY>

<!ATTLIST markfill
        scope       (document | sheet | page)  "page">

<!ELEMENT markover - o EMPTY>

<!ATTLIST markover
        attname     NAME                #REQUIRED
        attloc      CDATA               #IMPLIED>

<!ELEMENT marktext - o EMPTY>

<!ATTLIST marktext
        textid      NMTOKEN             #IMPLIED
        secrule     CDATA               #IMPLIED
        append      %yesorno            "0"
        unique      %yesorno            "0"
        appbefore   CDATA               #IMPLIED
        appsep      CDATA               #IMPLIED
        appafter    CDATA               #IMPLIED
        sort        %yesorno            "0">

<!-- This section describes the layout geometry of pages.
Multiple descriptions can be set up and referenced through the id
associated with the page model. -->
```

APPENDIX B

```
<!ELEMENT pagedesc  - o  (pageset)+>

<!ELEMENT pageset   - o  (rectopg, versopg?, rectobb?, versobf?, blankpg?)+>

<!ATTLIST pageset
        id          NMTOKEN -- an id --           #REQUIRED
        rectver     %yesorno -- toggle --         "1"
        blankpg     %yesorno -- toggle --         "1"
        orient      (portrait | landscape)        "portrait"
        mediainfo   CDATA                         #IMPLIED>

<!ELEMENT (rectopg | versopg)
                    - o  (pageres?, (pagespec | (pageref, topmarg?, botmarg?,
                          leftmarg?, rtmarg?, header?,
                          footer?, flowtext*, bordspec*))) -(span)>

<!ELEMENT blankpg   - o  (pageres?, (pagespec | (pageref, topmarg?, botmarg?,
                          leftmarg?, rtmarg?, header?,
                          footer?, flowtext*, bordspec*)),
                          (puttext | usetext | putgraph | ruling)*) -(span)>

<!ATTLIST (rectopg | versopg | blankpg)
        width       CDATA -- a size --                #IMPLIED
        nomdepth    CDATA -- a size --                #IMPLIED
        bind        (lleft |  ttop |  bbottom)        "lleft"
        chgmkwid    CDATA -- a size --                #IMPLIED
        chgmkoff    CDATA -- a size --                #IMPLIED
        chgmkplc    (pleft | pright | pin | pout | plftrt)  "pleft"
        topfloat    IDREFS -- idrefs of floatlocs --  #IMPLIED
        botfloat    IDREFS -- idrefs of floatlocs --  #IMPLIED
        mediainfo   CDATA -- a string --              #IMPLIED
        maxfloatpct NUMBER -- an integer (percentage) -- #IMPLIED>

<!ELEMENT (rectobb | versobf)
                    - o  (pageres?, header?, footer?)  -(span)>

<!ELEMENT pageres   - o  (reset | enumerat | savetext)*>

<!ELEMENT pagespec  - o  (topmarg, botmarg, leftmarg, rtmarg,
                          header, footer, flowtext+, bordspec*)>

<!ATTLIST pagespec
        pgid        NMTOKEN -- an ID --   #IMPLIED>

<!ELEMENT pageref   - o  EMPTY>
```

```
<!ATTLIST pageref
        pgidref     CDATA -- IDREF: reference to pgid --  #IMPLIED>

<!ELEMENT topmarg   - o  EMPTY>

<!ATTLIST topmarg
        nomdepth    CDATA -- a size --       #IMPLIED>

<!ELEMENT botmarg   - o  EMPTY>

<!ATTLIST botmarg
        nomdepth    CDATA -- a size --       #IMPLIED>

<!ELEMENT leftmarg  - o  EMPTY>

<!ATTLIST leftmarg
        width       CDATA -- a size --       #IMPLIED>

<!ELEMENT rtmarg    - o  EMPTY>

<!ATTLIST rtmarg
        width       CDATA -- a size --       #IMPLIED>

<!ELEMENT (header | footer)
                    - o (vquad | sectext | ruling | puttext | putgraph | usetext)*>

<!ATTLIST (header | footer)
        nomdepth    CDATA -- a size --       #IMPLIED
        maxdepth    CDATA -- a size --       #IMPLIED
        spaflow     CDATA -- a size --       #IMPLIED>

<!ELEMENT sectext   - o  (subchars)              +(vquad)>

<!ATTLIST sectext
        scope       (page | sheet | document)    #IMPLIED
        secref      IDREF                    #IMPLIED>

<!ELEMENT vquad     - o  EMPTY>

<!ATTLIST vquad
        verquad     (top | middle | bottom)  #IMPLIED>

<!ELEMENT flowtext  - o  (column, presp?, postsp?, gutter?, footnote?)>

<!ATTLIST flowtext
        numcols     (1 | 2 | 3 | 4 | 5 | 6 | 7 | 8)    #IMPLIED
```

```
        balance     %yesorno  -- toggle --             #IMPLIED
        topfloat    IDREFS -- idrefs of floatlocs --   #IMPLIED
        botfloat    IDREFS -- idrefs of floatlocs --   #IMPLIED
        chgmkplc    (pleft | pright | pin | pout | plftrt)  #IMPLIED>

<!ELEMENT column    - o  EMPTY>

<!ATTLIST column
        width       CDATA -- a size --                 #IMPLIED
        tolerance   CDATA -- a size --                 #IMPLIED
        vjprior     (col | comp)                       "col"
        topfloat    IDREFS -- idrefs of floatlocs --   #IMPLIED
        botfloat    IDREFS -- idrefs of floatlocs --   #IMPLIED>

<!ELEMENT gutter    - o  EMPTY>

<!ATTLIST gutter
        rlthick     CDATA -- a size --                   #IMPLIED
        ruleclr     (black | white | red |  orange | yellow |
                     green | blue | violet | brown | gray)  #IMPLIED
        rulepct     NUMBER -- an integer (percentage) -- #IMPLIED>

<!ELEMENT footnote  - o  (subchars?)>

<!ATTLIST footnote
        width       (col | flowtext)         "col"
        maxdepth    CDATA -- a size --       #IMPLIED
        ftnsepth    CDATA -- a size --       #IMPLIED
        ftnsepln    CDATA -- a size --       #IMPLIED
        ftnbrk      %yesorno -- a toggle --  "1"
        ftncntsp    CDATA -- a size --       #IMPLIED
        ftnconsl    CDATA -- a size --       #IMPLIED
        ftnconst    CDATA -- continue string -- #IMPLIED
        spabove     CDATA -- a size --       #IMPLIED
        ftnfloat    %yesorno -- a toggle --  "0">

<!ELEMENT bordspec  - o  EMPTY>

<!ATTLIST bordspec
        bordname    NMTOKEN            #REQUIRED
        bordent     ENTITY             #REQUIRED
        precedence  NUMBER             #IMPLIED>

<!-- This section describes the style characteristics associated
with all elements that are not a graphic or table. -->
```

APPENDIX B

```
<!ELEMENT styldesc  - o  (charsubset*, docdesc?, envdesc*, e-i-c, (e-i-c|charsubset*)
(tabatts | tgroupatts | colatts | subsetatts | rowatts | cellatts)>

<!ELEMENT docdesc   - o  (charlist)          -(subchars)>

<!ELEMENT envdesc   - o  (charlist)          -(subchars)>

<!ATTLIST envdesc
        envid       NMTOKEN  -- an id --  #REQUIRED>

<!ELEMENT e-i-c     - o  (charlist, (att|attif)*)>

<!ATTLIST e-i-c
        gi          NAMES -- a list of generic identifiers -- #REQUIRED
        context     CDATA -- see Context Syntax --            #IMPLIED
        occur       (only | first | last | middle | notlast | notfirst | all) "all"
        gitype      (element | pi) -- is eic for elem or pi? --  "element">

<!ELEMENT charsubset - o    (font?, leading?, hyphen?, wordsp?, sentxsp?,
    lettersp?, indent?, quadding?, highlt?, chgmark?, presp?, postsp?, keeps?,
    vjinfo?, textbrk?, span?, border?, float?, algroup?, suppress?, boxing?,
    link?, linkproc?, tabatts?, tgroupatts?, colatts*, subsetatts?, rowatts?,
    cellatts?, (reset | enumerat | ruling | puttext | putgraph | savetext |
    usetext)*)>

<!ATTLIST charsubset
        charsubsetid  NMTOKEN -- an id -- #IMPLIED
        charsubsetref CDATA -- IDREFS: list of referenced charsubset ids --
                                        #IMPLIED>

<!ELEMENT charlist  - o   (font?, leading?, hyphen?, wordsp?, sentxsp?,
        lettersp?, indent?, quadding?, highlt?, chgmark?, presp?, postsp?,
        keeps?, vjinfo?, textbrk?, span?, border?, float?, algroup?, suppress?,
        boxing?, link?, linkproc?, tabatts?, tgroupatts?, colatts*, subsetatts?,
        rowatts?, cellatts?, (reset | enumerat | ruling | puttext | putgraph |
        savetext | usetext)*)>

<!ATTLIST charlist
        envname     CDATA -- IDREF: reference to envid --     #IMPLIED
        inherit     %yesorno                            "0"
        charsubsetref CDATA -- IDREFS: list of referenced charsubset ids --
                                                            #IMPLIED>

<!ELEMENT att       - -  ((specval | fillval)*, charsubset?)>

<!ELEMENT attif     - -  ((specval | fillval)+, charsubset?, attelseif*, attelse?)>
```

```
<!ELEMENT attelseif - -  ((specval | fillval)+, charsubset?)>

<!ELEMENT attelse   - -  (charsubset)>

<!ATTLIST (att|attif|attelseif)
        logic       (and | or)     'and'>

<!ELEMENT specval   - o  EMPTY>

<!ATTLIST specval
        attname     NAME           #REQUIRED
        attloc      CDATA          #IMPLIED
        attval      CDATA          #REQUIRED>

<!ELEMENT fillval   - o  EMPTY>

<!ATTLIST fillval
        attname     NAME           #REQUIRED
        attloc      CDATA          #IMPLIED
        fillcat     CDATA          #REQUIRED
        fillchar    CDATA          #REQUIRED
        conrule     CDATA          #IMPLIED>

<!ELEMENT font      - o  EMPTY>

<!ATTLIST font
        inherit     %yesorno -- toggle --         #IMPLIED
        style       (serif | sanserif | monoser | monosans)  #IMPLIED
        famname     CDATA -- a font name --       #IMPLIED
        size        CDATA -- a size --            #IMPLIED
        posture     (upright |  oblique |  bsobl | italic | bsital)  #IMPLIED
        weight      (ultlight |  exlight |  light |  semlight |
                     medium | sembold |  bold |  exbold | ultbold)   #IMPLIED
        width       (ultcond |  excond |  cond |  semcond |
                     regular |  semexp | exp |  exexp |  ultexp)     #IMPLIED
        smallcap    %yesorno -- toggle --         #IMPLIED
        offset      CDATA -- size --              #IMPLIED>

<!ELEMENT leading   - o  EMPTY>

<!ATTLIST leading
        inherit     %yesorno -- toggle --    #IMPLIED
        lead        CDATA -- a size --       #IMPLIED
        force       %yesorno -- a toggle --  #IMPLIED>
```

```
<!ELEMENT hyphen    - o  EMPTY>

<!ATTLIST hyphen
        inherit     %yesorno -- toggle --         #IMPLIED
        lang        NMTOKEN -- IDREF: reference to a
                             hyphrule language -- #IMPLIED
        hyph        %yesorno -- toggle --         #IMPLIED
        zone        CDATA -- a size --            #IMPLIED>

<!ELEMENT wordsp    - o  EMPTY>

<!ATTLIST wordsp
        inherit     %yesorno -- toggle --    #IMPLIED
        minimum     CDATA -- a size --       #IMPLIED
        nominal     CDATA -- a size --       #IMPLIED
        maximum     CDATA -- a size --       #IMPLIED>

<!ELEMENT sentxsp   - o  EMPTY>

<!ATTLIST sentxsp
        inherit     %yesorno -- toggle --    #IMPLIED
        minimum     CDATA -- a size --       #IMPLIED
        nominal     CDATA -- a size --       #IMPLIED
        maximum     CDATA -- a size --       #IMPLIED>

<!ELEMENT lettersp  - o     EMPTY>

<!ATTLIST lettersp
        inherit     %yesorno -- toggle --           #IMPLIED
        minimum     CDATA -- a size --              #IMPLIED
        nominal     CDATA -- a size --              #IMPLIED
        maximum     CDATA -- a size --              #IMPLIED
        kerntype    (none |  pair |  track |
                     sector | pairtrk | trksectr)   #IMPLIED
        kernpair    ENTITY -- pointer --            #IMPLIED>

<!ELEMENT indent    - o     EMPTY>

<!ATTLIST indent
        inherit     %yesorno -- toggle --           #IMPLIED
        leftind     CDATA -- see Indent Syntax --   #IMPLIED
        rightind    CDATA -- see Indent Syntax --   #IMPLIED
        firstln     CDATA -- see Indent Syntax --   #IMPLIED>

<!ELEMENT quadding  - o     EMPTY>
```

```
<!ATTLIST quadding
        inherit     %yesorno -- toggle --             #IMPLIED
        quad        (right | left | center | in |
                     out | justify | asis)            #IMPLIED
        lastquad    (lright | lleft | lcenter | lin |
                     lout | ljustify | relative)      #IMPLIED>

<!ELEMENT highlt    - o     EMPTY>

<!ATTLIST highlt
        inherit     %yesorno -- toggle --           #IMPLIED
        reverse     %yesorno -- toggle --           #IMPLIED
        scoring     NUMBER -- an integer --         #IMPLIED
        scorewt     CDATA -- a size --              #IMPLIED
        scoreoff    CDATA -- a size --              #IMPLIED
        scorechron  %yesorno -- toggle --           #IMPLIED
        scorechr    CDATA -- a string --            #IMPLIED
        bckclr      (bblack | bwhite | bred | borange | byellow |
                     bgreen | bblue | bviolet | bbrown | bgray) #IMPLIED
        fontclr     (black | white | red | orange | yellow |
                     green | blue | violet | brown | gray) #IMPLIED
        bckpct      NUMBER -- an integer (percentage) -- #IMPLIED
        forpct      NUMBER -- an integer (percentage) -- #IMPLIED
        allcap      %yesorno -- toggle --           #IMPLIED
        scorespc    %yesorno -- toggle --           #IMPLIED>

<!ELEMENT chgmark   - -     (font?, indent?, quadding?, highlt?)>

<!ATTLIST chgmark
        literal     CDATA -- a string --            #IMPLIED
        barthick    CDATA -- a size --              #IMPLIED
        baroffset   CDATA -- a size --              #IMPLIED
        join        %yesorno -- a toggle --         #IMPLIED
        type        (content | start | end)         #IMPLIED
        cmclass     NMTOKEN -- a name --            #IMPLIED>

<!ELEMENT (presp | postsp)
                    - o     EMPTY>

<!ATTLIST (presp | postsp)
        minimum     CDATA -- a size --              #IMPLIED
        nominal     CDATA -- a size --              #IMPLIED
        maximum     CDATA -- a size --              #IMPLIED
        condit      (keep |  discard)               #IMPLIED
        priority    (force |  high |  med |  low |  none) #IMPLIED>
```

```
<!ELEMENT keeps     - o     EMPTY>

<!ATTLIST keeps
        keep        %yesorno -- toggle --           #IMPLIED
        scope       (col | page | line)             #IMPLIED
        widowct     NUMBER -- an integer --         #IMPLIED
        orphanct    NUMBER -- an integer --         #IMPLIED
        next        %yesorno -- toggle --           #IMPLIED
        prev        %yesorno -- toggle --           #IMPLIED
        floatsout   NMTOKENS -- IDREFS: idrefs of floatlocs -- #IMPLIED>

<!ELEMENT vjinfo    - o     EMPTY>

<!ATTLIST vjinfo
        inherit     %yesorno -- toggle --                   #IMPLIED
        presppr     (force |  high |  med |  low | none)    #IMPLIED
        postsppr    (pforce | phigh | pmed | plow | pnone) #IMPLIED
        keepspr     (kforce | khigh | kmed | klow | knone) #IMPLIED>

<!ELEMENT textbrk   - o     EMPTY>

<!ATTLIST textbrk
        startcol    %yesorno -- toggle --                   #IMPLIED
        startpg     (off | verso | recto | next)            #IMPLIED
        resumepg    (rverso | rrecto | rnext)               #IMPLIED
        pageid      CDATA -- IDREF: reference to pageset id -- #IMPLIED
        newpgmdl    (none | global | local)                 #IMPLIED
        startln     %yesorno -- toggle --                   #IMPLIED
        endln       %yesorno -- toggle --                   #IMPLIED>

<!ELEMENT span      - o     EMPTY>

<!ATTLIST span
        span        NUMBER -- an integer --       #IMPLIED>

<!ELEMENT border    - o     EMPTY>

<!ATTLIST border
        bordname    NMTOKEN -- a border name defined in bordspec -- #IMPLIED>

<!ELEMENT float     - o     EMPTY>

<!ATTLIST float
        flidref     NMTOKEN -- IDREF: to a float location -- #IMPLIED
        width       (page | col)                            #IMPLIED
        widowht     CDATA -- a size --                      #IMPLIED
```

```
        orphanht    CDATA -- a size --                        #IMPLIED
        scope       NAMES -- names of elements --             #IMPLIED
        pagetype    (same | facing | frontback | next | forward |
                                afterref | inline)           #IMPLIED
        inline      (never | unbroken | unsplit | unframed)  #IMPLIED
        multirefname NMTOKEN -- any token --                  #IMPLIED>

<!ELEMENT algroup   - o     EMPTY>

<!ATTLIST algroup
        refpoint    (top, first, middle, last, bottom) #IMPLIED
        postspace   %yesorno -- toggle --              #IMPLIED>

<!ELEMENT suppress  - o     EMPTY>

<!ATTLIST suppress
        sup         NUMBER -- an integer --  #IMPLIED>

<!ELEMENT boxing    - o     EMPTY>

<!ATTLIST boxing
        toffset     CDATA -- a size --                          #IMPLIED
        boffset     CDATA -- a size --                          #IMPLIED
        loffset     CDATA -- a size --                          #IMPLIED
        roffset     CDATA -- a size --                          #IMPLIED
        trel        (top | first)                               #IMPLIED
        brel        (last | bottom)                             #IMPLIED
        siderel     (text | col | content)                      #IMPLIED
        leftgap     CDATA -- a size --                          #IMPLIED
        rightgap    CDATA -- a size --                          #IMPLIED
        thick       CDATA -- a size --                          #IMPLIED
        ttype       (tblank | tsingle | tbold | tdouble |
                     ttriple | tdot | tdash)                    #IMPLIED
        btype       (bblank | bsingle | bbold | bdouble |
                     btriple | bdot | bdash)                    #IMPLIED
        ltype       (lblank | lsingle | lbold | ldouble |
                     ltriple | ldot | ldash)                    #IMPLIED
        rtype       (rblank | rsingle | rbold | rdouble |
                     rtriple | rdot | rdash)                    #IMPLIED
        inclr       (iblack | iwhite | ired | iorange | iyellow |
                     igreen | iblue | iviolet | ibrown | igray) #IMPLIED
        inpct       NUMBER -- an integer (percentage) --        #IMPLIED
        outclr      (black | white | red |  orange | yellow |
                     green | blue | violet | brown | gray)     #IMPLIED
        outpct      NUMBER -- an integer (percentage) --        #IMPLIED>
```

APPENDIX B

```
<!ELEMENT link      - o     EMPTY>

<!ATTLIST link
        sysid       CDATA                                   #IMPLIED
        targdocent  ENTITY                                  #IMPLIED
        targid      NAME                                    #IMPLIED
        endtargid   NAME                                    #IMPLIED
        linktype    (unspec|goto|newview)                   #IMPLIED
        uselink     (none|currdoc|sysid|targdoc)            #IMPLIED
        usetargid   %yesorno -- toggle --                   #IMPLIED
        useendtargid   %yesorno -- toggle --                #IMPLIED>


<!ELEMENT linkproc  - o     EMPTY>

<!ATTLIST linkproc
        loprocess   ENTITY                                  #IMPLIED
        exloproc    %yesorno -- toggle --                   #IMPLIED
        loconrule   CDATA -- a string --                    #IMPLIED
        liprocess   ENTITY                                  #IMPLIED
        exliproc    %yesorno -- toggle --                   #IMPLIED
        liconrule   CDATA -- a string --                    #IMPLIED>

<!ELEMENT reset     - o     EMPTY>

<!ATTLIST reset
        resetlist   NMTOKENS -- list of ids of
                                variables to be reset --   #IMPLIED>

<!ELEMENT enumerat  - o     EMPTY>

<!ATTLIST enumerat
        increm      NUMBER -- an integer --                 #IMPLIED
        enumid      CDATA -- an idref --                    #IMPLIED
        setvalue    %yesorno -- a toggle --                 #IMPLIED>

<!ELEMENT ruling    - -   (leading?, indent?, quadding?, presp?,
                           postsp?, keeps?, textbrk?, span?)>

<!ATTLIST ruling
        thick       CDATA -- a size --              #IMPLIED
        lentype     (rel | spec)                    #IMPLIED
        speclen     CDATA -- a size --              #IMPLIED
        rellen      (text | col)                    #IMPLIED
        voffset     CDATA -- a size --              #IMPLIED
        placemnt    (before | after)                #IMPLIED
```

```
        ruleclr     (black | white | red | orange | yellow |
                     green | blue | violet | brown | gray) #IMPLIED
        rulepct     NUMBER -- an integer (percentage) -- #IMPLIED
        type        (blank | single | bold |
                     double | triple | dot | dash)   #IMPLIED>

<!ELEMENT puttext   - -     (subchars?)>

<!ATTLIST puttext
        literal     CDATA -- a string --                    #IMPLIED
        placemnt    (before | after)                        #IMPLIED>

<!ELEMENT putgraph  - -     (subchars?)>

<!ATTLIST putgraph
        graphname   ENTITY -- an entity reference --        #REQUIRED
        width       CDATA -- a size --                      #IMPLIED
        depth       CDATA -- a size --                      #IMPLIED
        placemnt    (before | after)                        #IMPLIED
        scalefit    %yesorno -- a toggle --                 #IMPLIED
        hscale      NUMBER -- an integer (percentage) --    #IMPLIED
        vscale      NUMBER -- an integer (percentage) --    #IMPLIED
        hoffset     CDATA -- a size --                      #IMPLIED
        voffset     CDATA -- a size --                      #IMPLIED
        rotation    %yesorno -- a toggle --                 #IMPLIED>

<!ELEMENT savetext  - o     EMPTY>

<!ATTLIST savetext
        textid      CDATA   -- an id --                     #IMPLIED
        conrule     CDATA -- a string --                    #IMPLIED
        placemnt    (before | after)                        #IMPLIED
        append      %yesorno -- a toggle --                 #IMPLIED>

<!ELEMENT usetext   - -     (subchars?)>

<!ATTLIST usetext
        source      CDATA -- a string --            #IMPLIED
        placemnt    (before | after)                #IMPLIED
        userule     NMTOKEN -- an integer --        #IMPLIED
        useparam    CDATA -- a string --            #IMPLIED>

<!ELEMENT subchars  - o   (font?, leading?, hyphen?, wordsp?, sentxsp?,
        lettersp?, indent?, quadding?, highlt?, chgmark?, presp?, postsp?,
        keeps?, vjinfo?, textbrk?, span?, border?, float?, algroup?, boxing?,
        link?, linkproc?,
```

```
        tabatts?, tgroupatts?, colatts*, subsetatts?, rowatts?, cellatts?,
        (reset | enumerat | ruling | savetext)* )>

<!ATTLIST subchars
        charsubsetref CDATA -- IDREFS: list of referenced
                               charsubset ids -- #IMPLIED>

<!-- This section describes the characteristics associated with a table. -->

<!ELEMENT tabdesc   - o     (e-i-c+)>

<!ELEMENT tabatts   - o   (stdcellatts?)>

<!ATTLIST tabatts
        widthtype   (specific | relative)          #IMPLIED
        specwidth   CDATA -- a size --             #IMPLIED
        relwidth    NUMBER -- an integer (percentage) --     #IMPLIED
        frame       (all | top |  bottom | topbot | sides | none) #IMPLIED
        thick       CDATA -- a size --             #IMPLIED
        framestyle  (blank | single | bold |
                     double | triple | dot | dash) #IMPLIED
        consep      %yesorno  -- toggle --         #IMPLIED>

<!ELEMENT stdcellatts  - o   (charlist?)
              -(tabatts| tgroupatts| colatts| subsetatts| rowatts| cellatts)>

<!ATTLIST  stdcellatts
        colsepon    %yesorno -- toggle --          #IMPLIED
        rowsepon    %yesorno -- toggle --          #IMPLIED
        colsep      CDATA  -- a size --            #IMPLIED
        rowsep      CDATA  -- a size --            #IMPLIED
        colsepstyle (cblank | csingle | cbold |
                     cdouble | ctriple | cdot | cdash) #IMPLIED
        rowsepstyle (rblank | rsingle | rbold |
                     rdouble | rtriple | rdot | rdash) #IMPLIED
        leftmar     CDATA  -- a size --            #IMPLIED
        rightmar    CDATA  -- a size --            #IMPLIED
        topmar      CDATA  -- a size --            #IMPLIED
        botmar      CDATA  -- a size --            #IMPLIED
        halign      (right | left | center | justify | char)  #IMPLIED
        valign      (top | middle | bottom)        #IMPLIED
        char        CDATA -- see source syntax --  #IMPLIED
        charoff     NUMBER -- an integer (percentage) -- #IMPLIED
        reverse     %yesorno -- toggle --          #IMPLIED
        bckclr      (black | white | red | orange | yellow |
                     green | blue | violet | brown | gray) #IMPLIED
```

```
        shading     NUMBER -- an integer (percentage) --   #IMPLIED
        rotate      %yesorno -- a toggle --        #IMPLIED
        textwidth   CDATA -- a size --             #IMPLIED
        brkrow      %yesorno -- toggle --          #IMPLIED>

<!ELEMENT tgroupatts - o        (stdcellatts?)>

<!ATTLIST tgroupatts
        cols        NUMBER   -- integer --     #IMPLIED>

<!ELEMENT subsetatts - o        (stdcellatts?)>

<!ATTLIST subsetatts
        cols        NUMBER   -- number --      #IMPLIED
        keep        (col | page | none)        #IMPLIED
        subsettype  (head | body | foot)       #IMPLIED>

<!ELEMENT colatts   - o        (stdcellatts?)>

<!ATTLIST colatts
        colnum      CDATA  -- a string --               #IMPLIED
        colname     NMTOKEN -- an id --                 #IMPLIED
        colwidth    CDATA -- see column width syntax -- #IMPLIED
        spanname    NMTOKEN -- a name token --          #IMPLIED
        namest      NMTOKEN -- a name token --          #IMPLIED
        nameend     NMTOKEN -- a name token --          #IMPLIED>

<!ELEMENT rowatts   - o        (stdcellatts?)>

<!ELEMENT cellatts  - o        (stdcellatts?)>

<!ATTLIST cellatts
        colname     NMTOKEN -- a name token --    #IMPLIED
        spanname    NMTOKEN -- a name token --    #IMPLIED
        spandep     NUMBER -- integer --          #IMPLIED>

<!-- This section describes the characteristics associated with a
graphic inside a figure. -->

<!ELEMENT grphdesc  - o     (graphenv+, graphe-i-c+)>

<!ELEMENT graphenv  - o     (graphchars)>

<!ATTLIST graphenv
        graphenvid  NMTOKEN -- an id --    #IMPLIED>
```

APPENDIX B

```
<!ELEMENT graphe-i-c - o (e-i-c, (graphchars, gatt*)?)>

<!-- Note that for text block graphic objects, the charlist
on the e-i-c applies to composition of the text within the text
block bounding box. -->

<!ELEMENT graphchars - o (repro?, ((sizing, placemnt) | (textblock, placemnt))?)>

<!-- Note that to achieve multiple graphic objects in a
single reproduction area, no repro characteristics are specified
and the repro area of an ancestor of the graphic object is used.
-->

<!ATTLIST graphchars
        envname     CDATA -- IDREF: reference to graphenvid -- #IMPLIED>

<!ELEMENT repro     - o     EMPTY>

<!ATTLIST repro
        reprowid    CDATA -- size --                #IMPLIED
        reprodep    CDATA -- size --                #IMPLIED>

<!ELEMENT sizing    - o     EMPTY>
<!ATTLIST sizing
        graphname   ENTITY -- a pointer --                      #IMPLIED
        hscale      NUMBER -- an integer (percentage) --        #IMPLIED
        vscale      NUMBER -- an integer (percentage) --        #IMPLIED
        scalefit    %yesorno -- toggle --                       #IMPLIED
        llcordra    CDATA  -- a string of world coordinates -- #IMPLIED
        urcordra    CDATA  -- a string of world coordinates -- #IMPLIED>

<!ELEMENT textblock - o     EMPTY>
<!ATTLIST textblock
        tbwidth     CDATA -- size --                #IMPLIED
        tbdepth     CDATA -- size --                #IMPLIED
        horrefpt    (left | right)                  #IMPLIED
        verrefpt    (top | bottom)                  #IMPLIED>

<!-- Generally, the tbdepth would be omitted which would
imply that the textblock would have a depth that varies according
to the content (much like table cells generally work).

Note that the content of the e-i-c instance for text block
graphic objects becomes the textual content of the textblock
object. -->
```

```
<!ELEMENT placemnt  - o     EMPTY>
<!ATTLIST placemnt
        hplace      (left | center | right | hnone)           #IMPLIED
        vrplace     (top | middle | bottom | none)            #IMPLIED
        coordst     CDATA -- a string of world coordinates -- #IMPLIED
        coordend    CDATA -- a string of world coordinates -- #IMPLIED
        rotation    %yesorno  -- toggle --                    "0">

<!-- Note that coordend is ignored for textblocks -->

<!ELEMENT gatt      - o ((specval | fillval)+, graphchars?)>
<!ATTLIST gatt        logic       (and | or)  'and'>

<!-- Note that charlist is not needed here as it applies
only to text block graphic objects and are handled within the
e-i-c portion of the graphe-i-c. -->

<!-- This section describes the characteristics associated with elements
     to be placed in the footnote area. -->

<!ELEMENT ftndesc   - o     (e-i-c, ftnatts)*>

<!ELEMENT ftnatts   - o     (charlist)      -(keeps,span)>

<!-- This section describes the information for the generation
     of change pages and related components.  -->

<!ELEMENT chngdesc - o (chnglevl?, lepdesc*, pgbrk?, chnginst?)>

<!ELEMENT chnglevl - o (lvlnames?, (chngtag | chngatt)*)>
<!ATTLIST chnglevl
        doctag          NAME                    #REQUIRED
        docattr         NAME                    #IMPLIED
        docname         NAME                    #IMPLIED
        pagename        NAME                    #IMPLIED
        leafname        NAME                    #IMPLIED
        sprdname        NAME                    #IMPLIED
        docnumb         NAME                    #IMPLIED
        pagenumb        NAME                    #IMPLIED
        leafnumb        NAME                    #IMPLIED
        sprdnumb        NAME                    #IMPLIED
        formatnm        NAME                    #IMPLIED
        formatlv        NAME                    #IMPLIED>

<!ELEMENT lvlnames - o EMPTY>
<!ATTLIST lvlnames
```

```
        cntnrtag        NAME                    #REQUIRED
        subcntnr        NAME                    #REQUIRED
        nametag         NAME                    #IMPLIED
        nameattr        NAME                    #IMPLIED
        numbtag         NAME                    #IMPLIED
        numbattr        NAME                    #IMPLIED
        levlzero        %yesorno                "0">

<!ELEMENT chngtag   - - (chngchar*)>

<!ATTLIST chngtag
        taglist         NAMES                   #REQUIRED
        levltype        (number | name)         "number"
        levlattr        NAME                    #REQUIRED
        chngtype        (insert | delete | change)      #IMPLIED
        typeattr        NAME                    #REQUIRED>

<!ELEMENT chngatt  - - (chngchar*)>
<!ATTLIST chngatt
        taglist         NAMES                   #IMPLIED
        attrlist        NAMES                   #REQUIRED
        chngtype        (insert | delete | change)      #REQUIRED
        attrcont        (level | switch)        "level"
        chlvtype        (name | number)         #IMPLIED>

<!ELEMENT chngchar - - (charlist, att*)>
<!ATTLIST chngchar
        when            (less | equal | greater)        "equal">

<!ELEMENT lepdesc  - o EMPTY>
<!ATTLIST lepdesc
        string          ID                                      #REQUIRED
        scope           NAME                                    #IMPLIED
        pgsets          IDREFS                                  #IMPLIED
        leptag          NAME                                    #REQUIRED
        lepline         NAME                                    #REQUIRED
        group           %yesorno                                "1"
        prefix          IDREF                                   #IMPLIED
        blankflg        %yesorno                                "1"
        deleted         %yesorno                                "1"
        groupblk        (no | yes | end | begin | endbegin)     "end"
        leaflevl        %yesorno                                "1"
        folio1          NAME                                    #REQUIRED
        folio2          NAME                                    #REQUIRED
        chngnumb        NAME                                    #IMPLIED
        chngname        NAME                                    #IMPLIED
```

```
        pgstatus        NAME                                    #IMPLIED
        single          NAME                                    #IMPLIED>

<!ELEMENT pgbrk - - (pgattr*)>

<!ATTLIST pgbrk
        tag             NAME            "pgbrk"
        place           (recto | rectoverso | verso)    "rectoverso">

<!ELEMENT pgattr - o EMPTY>
<!ATTLIST pgattr
        attrname        NAME            #REQUIRED
        attrtype        (folio | foliodesc | nextfolio | nextfoliodesc |
                         sequence | prefix | parabreak | hyphen | shrink |
                         rectoverso | change | changename |
                         security | empty | ignore) #REQUIRED>

<!ELEMENT chnginst - o EMPTY>
<!ATTLIST chnginst
        folio           NMTOKEN                                 #REQUIRED
        nxtfolio        NMTOKEN                                 #IMPLIED
        prefix          NMTOKEN                                 #REQUIRED
        seq             IDREF                                   #REQUIRED
        foliotag        NAME                                    #REQUIRED
        ptlevel         IDREFS                                  #IMPLIED
        pointtag        NAMES                                   #IMPLIED
        instleaf        (yes | no | perpgdesc)                  "perpgdesc">
]
```

B.6 GLOSSARY

B.6.1 EQUIVALENCY DEFINITIONS. The following values should be used for conversion purposes:

1 inch = 72 points +/- 0.4 percent

1 pica = 12 points

B.6.2 DEFINITION OF TERMS.

B.6.2.1 Attribute name. The name of an attribute as assigned in the attribute definition list declaration of a DTD.

B.6.2.2 Border. Graphic that appears along page boundaries.

B.6.2.3 Character fill. The repeated use of a character (possibly the space character), or string of characters, to occupy space within running text, for example, leader dots.

B.6.2.4 Characteristic. Specification of a formatting property for an element-in-context. The characteristics of text, graphics, and tables define how they are processed and output by the system.

B.6.2.5 Context. The lineage of ancestor elements of a particular element instance in a source document and its representation in a FOSI. Context is a part of element-in-context specification.

B.6.2.6 Coordinate, world system. System used to describe the two-dimensional space in which a graphic is defined. The world coordinate system is comprised of X and Y coordinates having a point of origin in the lower left corner (coordinates 0,0) and the upper right corner (coordinates 10000,10000).

B.6.2.7 Default. Characteristic values that are assigned to the document element or a named environment from which characteristic values can be derived if left unspecified and not inherited.

B.6.2.8 Document Type Definition (DTD). Rules that are required to apply SGML to document markup. The DTD includes a formal specification of the document element types, their relationships and attributes, and references that can be represented by markup. It therefore defines the vocabulary of the markup for which SGML defines the syntax. It is determined by the document's application. Specifically, the document type definition specifies:

a. The generic identifiers (GIs) of elements that are permissible in a document of this type.
b. For each GI, the possible attributes, their range of values, and defaults.
c. For each GI, the structure of its content, including:
    1. In which subelement GIs can occur, and in what order.
    2. Whether text characters can occur.
    3. Whether noncharacter data can occur.

B.6.2.9 Element-in-context (e-i-c). The complete set of information necessary to identify a particular set of instances of an element type within a document, including its generic identifier, context, and occurrence.

B.6.2.10 e-i-c. See element-in-context.

B.6.2.11 Em space. The approximate horizontal space taken up by a capital 'M'. It is often the same width as the current point size. The exact size of an em space is determined by the font in use.

B.6.2.12 Enumeration. The generation of ordering labels (for example, item counts) or other counting (for example, page numbering) by the composition system. The numbers that are to be generated by the system are identified as counters and may be combined with other counters and literals to form compound numbers.

B.6.2.13 Font. Collection of character images. A font is a complete set of characters in one design and size. A font system creates character sets that are derivative of character representations. Expanded, italic, and bold fonts are three different fonts and the medium font, that the derivatives are based on, is a different font.

B.6.2.14 Formatting Output Specification Instance (FOSI). The set of characteristics and values chosen from the Output Specification to represent the formatting requirements for a particular type of document. A FOSI is an SGML document conforming to the OS DTD.

B.6.2.15 Generic identifier (GI). Unique name that identifies and refers to an element type in a DTD. The GI is part of an element-in-context.

B.6.2.16 GI. See generic identifier.

B.6.2.17 Graphic placement. Horizontal and vertical placement of a graphic within a reproduction area. Graphic placement can be specified with relative values or coordinates.

B.6.2.18 Graphic sizing. Constraints on the size or view of graphics to be placed in reproduction area dimensions. Graphics can be scaled and cropped.

B.6.2.19 Highlight. Special presentation characteristics for text, including scoring, color, and screens.

B.6.2.20 Hyphenation. Breaking a word at the end of a line of text for purposes of line justification.

B.6.2.21 Identifier (ID). An identifier that is unique throughout a document.

B.6.2.22 Identifier reference (IDREF). A reference to an ID.

B.6.2.23 Indent. Positioning in the writing (horizontal) direction. Indents can be relative to area boundaries or margins set by other indents.

B.6.2.24 Inheritance. Mechanism whereby unspecified characteristic values are derived from the corresponding values assigned to the parent element.

B.6.2.25 Justification, vertical. The positioning of text lines in the vertical direction for purposes of page filling and column balancing.

B.6.2.26 Keeps. Conditions for controlling or disallowing breaking elements over line, column, or page boundaries.

B.6.2.27 Layout areas. Rectangular areas on a page in which output stream content is placed.

B.6.2.28 Leading. The distance between the baseline of text lines in a single element.

B.6.2.29 Letterspace. Space between letters in a word that may be adjusted for purposes of line justification.

B.6.2.30 List, finite. The set of choices for the value of a particular characteristic.

B.6.2.31 Margin, bind. Area of a page designated for binding, that is, physically connecting pages with a staple, stitch, or glue. Typically, the bind margin of a page is larger than the opposite margin to allow for the space taken up by binding. The bind margin also serves as a reference for quadding of text that is toward the inside of the book (in) or toward the outside of the book (out).

B.6.2.32 Occurrence. Order of appearance of an element in relation to other like elements.

B.6.2.33 Output Specification (OS). Guidelines for the interchange of style and formatting information of technical documents between all types of publishing systems. The OS provides the range of possible ways of describing formatting requirements.

B.6.2.34 Page fidelity. Preservation of the exact presentation characteristics and the exact information on pages exchanged between systems.

B.6.2.35 Page integrity. Preservation of the exact information on each page as it is exchanged between systems. Page integrity does not guarantee that information will be presented on the page in exactly the same way or appearance after exchange between systems (page fidelity).

B.6.2.36 Page model. Collection of pagination characteristics describing the layout areas that occur on a page.

B.6.2.37 Page set. A page model that is used for a set of like pages that vary only when they are recto, verso, or blank.

B.6.2.38 Parseability, machine. The ability to verify, through a computer program, that a FOSI has correct syntax (conforms to the OS DTD).

B.6.2.39 Point. Unit of measurement approximately equal to 1/72 inch.

B.6.2.40 Pointer. A reference to information contained in an external file. Pointers behave like SGML external entity references.

B.6.2.41 Postspace. Amount of space in the vertical direction added after content (in addition to its leading).

B.6.2.42 Prespace. Amount of space in the vertical direction added before content (in addition to its leading).

B.6.2.43 Priority. Specification of which values take precedence when there is a conflict. The priority characteristic range of values is “Force”, “High”, “Medium”, “Low”, and

"None". Force has the greatest priority value and none has the least. The value is compared to provide a solution to the conflict. Where priority characteristic values are equal, there are additional precedence rules that are applied depending on where the characteristic is encountered.

B.6.2.44 Puttext. Describes system-generated text that appears in the output data stream.

B.6.2.45 Pseudo-element. A construct that can act as a reference to an e-i-c entry in the FOSI.

B.6.2.46 Quadding, horizontal. Horizontal alignment of text lines within an element with respect to the layout area boundaries.

B.6.2.47 Quadding, vertical. Vertical alignment of an element with respect to area boundaries.

B.6.2.48 Reproduction area dimensions. The width and depth of an area that is to be filled with a graphic.

B.6.2.49 Ruling. Drawing of a line segment.

B.6.2.50 Savetext. Text to be saved for use elsewhere in a document.

B.6.2.51 Span. Positioning of content across all columns on a page.

B.6.2.52 String. Literal sequence of characters, including alpha, numeric, and special (pi) characters.

B.6.2.53 Textbreak. Characteristics for controlled interruptions of normal text flow, including starting text at the beginning of a column, page, or line.

B.6.2.54 Toggle. A value type treated as a Boolean variable where the only distinction between values is zero ("off") and non-zero ("on").

B.6.2.55 Usetext. Describes the use of content that has been saved with the Savetext characteristic.

B.6.2.56 Wildcard. Specification of zero or more levels of ancestry in a context that are not called out by name.

B.6.2.57 Wordspace. Space between words in a text line that may be adjusted for purposes of line justification.

B.6.3 ACRONYMS.

CCITT	Consultative Committee on International Telegraphy and Telephony (now the International Telecommunications Union Telecommunication Standardization Sector, ITU-T)
CGM	Computer Graphics Metafile
DTD	Document Type Definition
e-i-c	Element-in-context
FOSI	Formatting Output Specification Instance
GI	Generic Identifier

APPENDIX B

IGES	Initial Graphics Exchange Specification
ISO	International Organization for Standardization
OS	Output Specification
ITU-T	International Telecommunications Union Telecommunication Standardization Sector (previously the Consultative Committee on International Telegraphy and Telephony, CCITT)
WYSIWYG	What You See Is What You Get

CONCLUDING MATERIAL

Custodians:
Army – TM
Navy – OM
Air Force – 16

Preparing Activity:
DISA – DC3
(Project IPSC 0336)

Review activities:
OASD – DO1, DO7, IQ, IR
Army – AC, AL, AT, CR, MI, PT, SC1, SC3, TM1
Navy – AS, CG2, CG5, CH, EC, MC5, ND, NM, TD
Air Force – 13, 17, 19, 33, 79, 90, 93
DISA – DC1, DC5
DIA – DI1, DI3
NORAD & USSPACECOM – US
NSA – NS
Others – DOE, DOT-OST, GPO, NCS

STANDARDIZATION DOCUMENT IMPROVEMENT PROPOSAL

INSTRUCTIONS

1. The preparing activity must complete blocks 1, 2, 3, and 8. In block 1, both the document number and revision letter should be given.

2. The submitter of this form must complete blocks 4, 5, 6, and 7.

3. The preparing activity must provide a reply within 30 days from receipt of the form.

NOTE: This form may not be used to request copies of documents, nor to request waivers, or clarification of requirements on current contracts. Comments submitted on this form do not constitute or imply authorization to waive any portion of the referenced document(s) or to amend contractual requirements.

I RECOMMEND A CHANGE:	**1. DOCUMENT NUMBER** MIL-PRF-28001C	**2. DOCUMENT DATE** ***(YYYY-MM-DD)*** 1997-05-02

3. DOCUMENT TITLE MARKUP REQUIREMENTS AND GENERIC STYLE SPECIFICATION FOR EXCHANGE OF TEXT AND ITS PRESENTATION

4. NATURE OF CHANGE ***(Identify paragraph number and include proposed rewrite, if possible. Attach extra sheets as needed.)***

5. REASON FOR RECOMMENDATION

6. SUBMITTER

a. NAME *(Last, First, Middle Initial)*	b. ORGANIZATION	
c. ADDRESS *(Include Zip Code)*	d. TELEPHONE *(Include Area Code)* (1) Commercial (2) DSN *(If applicable)*	7. DATE SUBMITTED *(YYYY-MM-DD)*

8. PREPARING ACTIVITY

a. NAME CALS Digital Standards Office DISA Center For Standards Code JIEO/JEBEB	b. TELEPHONE *(Include Area Code)* (1) Commercial (703) 735-3568 (2) DSN 653-3568
c. ADDRESS *(Include Zip Code)* 10701 Parkridge Boulevard Reston, VA 20191-4357	IF YOU DO NOT RECEIVE A REPLY WITHIN 45 DAYS, CONTACT: Standardization Program Division 5203 Leesburg Pike, Suite 1403, Falls Church, VA 22041-3466 Telephone (703) 681-9340 DSN 761-9340

INSTRUCTIONS: In a continuing effort to make our standardization documents better, the DoD provides this form for use in submitting comments and suggestions for improvements. All users of military standardization documents are invited to provide suggestions. This form may be detached, folded along the lines indicated, taped along the loose edge (DO NOT STAPLE), and mailed. In block 4, be as specific as possible about particular problem areas such as wording which required interpretation, too rigid, restrictive, loose, ambiguous, or was incompatible, and give proposed wording changes which would alleviate the problems. Enter in block 5 any remarks not related to a specific paragraph of the document. If block 6 is filled out, an acknowledgement will be mailed to you within 30 days to let you know that your comments were received and are being considered.

NOTE: This form may not be used to request copies of documents, nor to request waivers, deviations, or clarification of specification requirements on current contracts. Comments submitted on this form do not constitute or imply authorization to waive any portion of the referenced document(s) or to amend contractual requirements.

(Fold along this line)

(Fold along this line)

CALS DIGITAL STANDARDS OFFICE
DISA CENTER FOR STANDARDS
CODE JIEO/JEBEB
10701 PARKRIDGE BOULEVARD
RESTON VA 20191-4357

CALS DIGITAL STANDARDS OFFICE
DISA CENTER FOR STANDARDS
CODE JIEO/JEBEB
10701 PARKRIDGE BOULEVARD
RESTON VA 20191-4357