OSF Documentation SIG

Recommendations for the OSF DTD

July 31, 1992

Version: OSF DTD Recommendations, Version 1.0

This document contains the Documentation Special Interest Group's recommendations to OSF for an SGML document type definition (DTD) to be used in OSF document interchange.

Contents

2 Overview of the Recommendations

3 Manual Organization Recommendations

B Handling Document Fragments

C Prototype of the Recommendations

Glossary

Examples

Figures

Tables

Preface

This document contains the Documentation Special Interest Group's recommendations to OSF for an SGML document type definition (DTD) to be used in OSF document interchange.

Availability

This document is available to any person or organization for study and use as a DTD specification. The OSF Documentation SIG is responsible for making it available and incorporating any updates and corrections. The document is available in hardcopy and online (PostScript) forms. The source code for the prototype DTD is also available.

To request the document or the DTD, or to comment on the document, contact Eve Maler (Digital Equipment Corporation, Nashua, NH, USA): telephone 603/881–0310, FAX 603/881–0310, or Internet address `maler@zk3.dec.com`. Implementation issues should be mailed to the `sig-doc-di@osf.org` distribution list.

Audience

This document is intended primarily for implementors, but also for documenters and users, of any SGML DTD or document-processing application that is related to these recommendations. Technical SGML concepts are not defined in this document; implementors are expected to know details of SGML application development.

Related Documents

See the following documents for more information:

- Historical design documents

  OSF members can get hardcopies of the historical documents comparing the DTD submissions and recording component recommendation rationales. To request a set, send your address to Jeanne El Andaloussi (Bull, Paris, France): telephone +33 1 4356 4478, FAX +33 1 4356 4492, or Internet address `J.Elandaloussi@frpr.bull.fr`.

- Position papers

  OSF members can get copies of the OSF Documentation SIG information architecture working group (IAWG) position papers and other working documents. The approved IAWG position papers are Scope, Language Design, Applications/Purposes, Localization and Internationalization, SGML Declarations and Parsers, and Standards and Interchange. The Graphics Notations Requirements paper is unapproved, but a valuable resource.

  For information on getting a copy of any of the IAWG working documents or the

OSF Request for DTDs (RFDTD), contact John Bowe (OSF, Cambridge, MA, USA): telephone 617/621–7269, FAX 617/225–2782, or Internet address `bowe@osf.org`.

- OSF rollout plans

  OSF members can also get copies of the OSF *DTD Documentation Tools Plan*, *DTD Documentation Migration Plan*, and *OSFDTD Project Plan* from John Bowe.

- Part 11 of ISO Technical Report 9573, *Techniques for SGML Application at the ISO Central Secretariat*,

  This document can be obtained from the Graphic Communications Association in Alexandria, Virginia. Part 11 will be revised soon.

Conventions Used in This Document

Names in **boldface type** are names of recommended components (both elements and attributes). The names correspond to those in the tree diagrams that appear throughout the document. Note that the names used in this document are descriptive only; they are not intended as recommendations for element generic identifiers, attribute names, or attribute token values.

Figure A-1 describes the notation used in the tree diagrams in this document. In various places in this document, a more informal version of this notation is used for descriptive purposes.

Acknowledgments

The editor of this document gratefully acknowledges Susanna Ries and Marjorie Morse of Digital Equipment Corporation for their help in its preparation.

Development of the Recommendations 1

This chapter describes how the recommendations were created and outlines the principles used to develop them.

1.1 History

As part of a plan to move to SGML for document interchange, the Open Software Foundation (OSF) issued a Request for DTDs (RFDTD) in December 1990. The RFDTD described the interchange needs of OSF, its technology suppliers, and its licensees, and outlined some requirements and goals for the DTD(s).

The RFDTD indicated that OSF planned to work with the Documentation Special Interest Group (SIG) to review all DTD submissions, select technology, and write a rationale for the final decision. The Documentation SIG's Information Architecture Working Group (IAWG) had helped OSF develop its original goals for SGML interchange through 1990; because of the extreme importance of achieving successful interchange with OSF, the IAWG has directed all its efforts since the issuance of the RFDTD to helping OSF choose SGML DTD technology.

The IAWG determined early on that selecting features from each submission without applying a cohesive set of design principles would result in an inconsistent, unusable DTD. It developed such principles, recorded them in a series of position papers, and formed a subgroup to analyze the submissions in depth and develop specific DTD design recommendations that adhere to the principles. The IAWG, and later the design group, met every six to eight weeks to perform this work. The result is this recommendation document.

The following people have participated in the design group at various times as official OSF or member company representatives:

Bull	Jeanne El Andaloussi	J.Elandaloussi@frpr.bull.fr
Digital	Eve Maler (group whip, editor of recommendations)	maler@zk3.dec.com
Hewlett-Packard	Richard Artz Dave Hollander Bill Kaster Manny Yousefi	richard@fc.hp.com dmh@hpasdd.hp.com wsk@hpclpa.hp.com manny@hpclpb.corp.hp.com
Hitachi	Nancy Paisner	nancy@hicomb.hi.com
IBM	Sharon Adler Jan Prusack Wayne Wohler	adlers@bldvm1.ibm.com janp@ibmpa.awdpa.ibm.com wohler@bldvm1.vnet.ibm.com
OSF	John Bowe	bowe@osf.org
Prime	Susan Windheim	
Santa Cruz Operation	Murray Maloney	murray@scocan.sco.com

The following additional people actively participated in this IAWG effort:

Digital	Lee Fogal (SIG chair) Kathy Greenleaf	`fogal@zk3.dec.com` `greenleaf@greeny.enet.dec.com`
HP	Dea Burmeister Alden Chang Joe Schneider	
IBM	Truly Donovan Dennis McCutchen (then IAWG chair)	 `dennis@ausvmq.vnet.ibm.com`
OSF	Elizabeth Connolly Fred Dalrymple	`beth@osf.org` `fred@osf.org`

1.2 Design Group Project Tasks

The design group analyzed the features of OSF's Semantic Markup Language (SML), the submissions, and other relevant markup systems as follows:

1. Compared the hierarchies of their semantic components by representing them as tree diagrams (recorded in the historical tree diagram document).
2. Compared their semantic distinctions by creating a matrix of component names and the corresponding markup form in each system.

   We added to the matrix any specific component requirements from additional IAWG and design group analysis.

To make the recommendations, the design group then took the following steps using the design principles outlined in Section 1.3:

1. Determined in a preliminary fashion which semantic components represented in the matrix should be included in the OSF DTD(s) (summarized in the historical matrix, and detailed in the historical component decision rationale).
2. Developed complete requirements for the allowable contexts, content models, and other design features of the semantic components (recorded in this document).

Note that the group did not pass judgment on the submissions.

Note

The historical documents contain information on specific submission features and are therefore available only to OSF members. Because they reflect our analysis of the submissions, they do not correspond exactly to the further synthesis work that is recorded in this document.

However, these documents give perspective on how the design work proceeded; we strongly recommend consulting them for background information on particular recommendations.

1.3 Design Principles

The design group applied various principles in developing its recommendations. Many of the principles were encoded by the RFDTD and by the IAWG in a series of position papers: Scope, Language Design, Applications/Purposes, Localization and

Internationalization, SGML Declarations and Parsers, and Standards and Interchange. (The RFDTD and papers are referred to in this section, but are not reproduced in this document.) Other principles were developed during the IAWG and design group analysis process.

Table 1-1 summarizes the design principles, which are discussed in the following sections.

Table 1-1: Design Principles

Principle Name	Description
Scope	Accommodate software guides and reference manuals, reference pages, and integrated hardware/software manuals.
Interchange	Develop an interchange data protocol that ensures usable, high-quality documents to the extent possible; place the burdens of conformance and validation on sender at each stage.
Data integrity	Allow information to be encoded abstractly and with a high degree of semantic precision, and make relationships among components explicit.
Presentation independence	Remove or recast presentational information, or encode it in attributes.
Document model viability	Adhere to assumptions that readers make about the document types.
Component ordering	Choose consistent, company-neutral, orderings of information.
Planning for DTD evolution	Given the need for functional DTD additions, plan for such additions and define their scope carefully.
Markup consistency	Meet writer expectations by making the content models and attributes of like elements similar.
Augmentation	Allow for augmentation of navigation information.
Linking	Prepare for hypermedia by providing a robust set of linking components.
Localization	Provide semantic components that aid translation and localization.
CFCM	Agree on a common DTD for reference pages with the Committee for Common Man while accounting for OSF interchange needs.
Availability	Make the OSF DTD available for public study and use.

1.3.1 Document Type Scope

According to the Scope position paper, the DTD or DTDs that are developed need to encompass the following document types:

- Software guides and reference manuals
- Reference pages
- Integrated hardware/software manuals

The design group has recommended a single DTD that accounts for all these document types; the top-level element corresponds to a single logical document. (We have assumed that processing and management of multiple-document sets and collections will be performed without an SGML encoding for such.)

The group ignored the following tasks as being outside the scope of our work:

- Creation and maintenance of variants of the interchange DTD for in-house use in creating and editing documents, an Applications/Purposes principle, or for use outside OSF interchange with different languages and cultures

  However, see Appendix B for information on draft development that might be relevant to the interchange DTD.
- Minimization design and methods of normalizing instances in preparation for interchange (see Section 1.4.2 for a discussion of this principle)
- Names for element generic identifiers, attributes, attribute token values, and so on
- Making the DTD readable, an Applications/Purposes principle
- Handling multilingual documents (documents that contain the same information in multiple languages), a Localization and Internationalization principle

  We discovered that OSF had no need for such documents, and removed them from the scope
- Getting an external review of the recommendations to assess their relationship to the CALS DTD and other markup systems, a Standards and Interchange principle
- Using standard (for example, HyTime-compliant) linking applications, a longstanding IAWG discussion topic

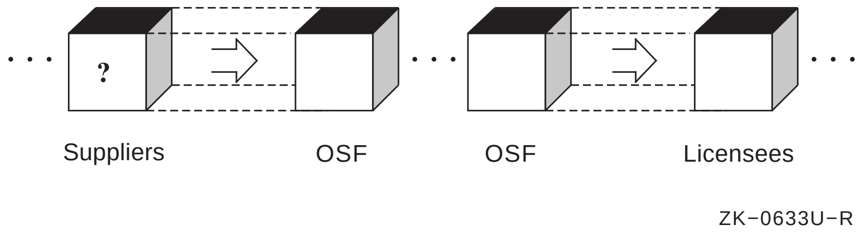
1.3.2 Interchange

According to the Applications/Purposes position paper, the DTD needs to facilitate successful document interchange among OSF, its technology suppliers, and its licensees.

Successful interchange can be characterized as the use of predictable, stable data protocols and processes that result in the exchange of high-quality, usable documentation. Exchanging documents is like sending data across a network in that a protocol must be negotiated and followed. In this case, the protocol consists of the rules that the transferred data must adhere to, such as the DTD, a standard SGML declaration, and any other constraints; the protocol defines the acceptable form of the data. (Protocols for data transport are outside the scope of this document.)

There are two stages at which SGML interchange is important for OSF and its interchange partners: supplier-to-OSF and OSF-to-licensee. Figure 1-1 illustrates the requirements for validity at these stages. The boxes represent a collection of document data that the sender has validated as conforming to a particular protocol.

Figure 1-1: Interchange Protocol Requirements

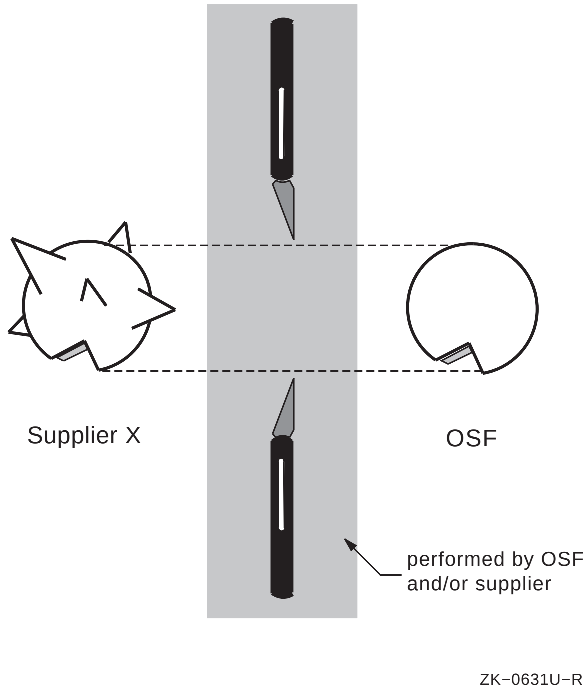


In the first stage, some number of technology suppliers sends documents or partial documents to OSF during the development of an OSF offering. The ratio of senders to receivers is possibly several-to-one; sometimes this stage can resemble a multiple-writer project, with free sharing of files back and forth, rather than a single instance of document interchange. OSF may want to negotiate a different protocol, for example different data forms (if the suppliers use many different data forms in-house) or not-yet-validated documents. The question mark represents OSF's option to agree with any supplier to use a different protocol.

In the second stage, OSF sends documents to licensees. The ratio of senders to receivers is one-to-many; it is impractical for OSF to define and adhere to multiple alternate protocols, and it is impractical for licensees to prepare their publication environments to handle more than one single data form coming from OSF. Therefore, all parties have a need to negotiate a single protocol, and the licensees have a strong need for it to reflect high standards so that the received information will be immediately usable for their own purposes.

Suppliers, OSF, and licensees have different responsibilities in each transaction. Figure 1-2 illustrates the typical responsibilities during the first stage of interchange. The spheres represent individual documents with various idiosyncrasies in their markup.

Figure 1-2: Responsibilities of Suppliers



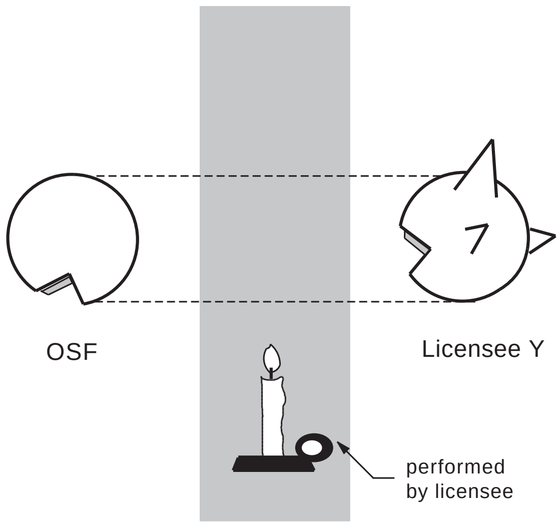
Assume Supplier X uses a DTD different from the OSF interchange DTD. The spikes on X's sphere represent semantic distinctions in the document that the X DTD makes but the OSF DTD does not. (For example, if the X DTD distinguishes between glossary definitions for natural-language words and those for programming keywords but the OSF DTD does not, instances of the two kinds of definitions in documents are ''spikes.'') The chips in X's sphere represent places in the document where the OSF DTD – not necessarily the X DTD – makes a semantic distinction, but the information is not marked up enough to identify the information as belonging in that class. (For example, if the OSF DTD has an element to contain command syntax but some of the syntax diagrams in the document are marked up only with miscellaneous lines display tags, these instances are ''chips.'')

Barring any negotiation that changes the terms of interchange, Supplier X is responsible for removing the spikes (a process represented by the knives). However, the effort to fill in chips is much more expensive than removing spikes. SGML validation parsing cannot detect chips; human inspection is probably needed. Therefore, some chips are expected to remain during interchange. Ultimately, X is responsible for validating that the document conforms to the protocol.

If the instance is minimized and OSF wants to normalize it, OSF is responsible for doing this. See Section 1.4.2 for more information on minimization.

Figure 1-3 illustrates the typical responsibilities during the second stage of interchange. Note that during the time OSF owns the document, it might have modified the data form of the document before returning it to interchange form.

Figure 1-3: Responsibilities of Licensees



Assume Licensee Y uses a DTD different from the OSF interchange DTD. If Y's environment is not set up to handle the OSF DTD, it is responsible for modifying the document to conform to its own requirements (a process represented by the candle); this adaptation can involve adding spikes, taking out chips, or making wholesale changes. OSF is responsible for validating that the document conforms to the protocol.

If the instance is minimized and the licensee wants to normalize it, the licensee is responsible for doing this. See Section 1.4.2 for more information on minimization.

Note that where this discussion mentions different DTDs, it could just as well mean different markup systems or other document-encoding systems entirely. In addition, interchange requirements that are not syntactic in nature (for example, accuracy and completeness standards) have not been addressed here in detail, but it is assumed that ''validation'' will also include human-directed processes (such as editing and technical review) that will ensure conformance and quality.

1.3.3 Data Integrity

According to the Language Design position paper, the DTD needs to have a high degree of data integrity, which means encoding information abstractly enough, and identifying semantic distinctions that are useful enough, to anticipate a wide variety of applications for the information over time.

To meet this requirement, the design group tended to recommend a large number of semantic distinctions, including all those from SML (as the RFDTD specifies). Where submissions had sets of related semantic distinctions in a particular domain, the group tried to define common components that met all the expressed needs, and where submissions had sets of unrelated semantic distinctions, the group tended to recommend a union of all the submitted components.

Where only one company had components in a semantic domain, we questioned whether this type of information would appear in OSF interchange. In general, we declared non-OSF-related components to be spikes and did not recommend them. However, sometimes other companies felt the distinctions were useful or even had similar components in the non-submitted portions of their markup, and in many of these cases the group agreed to recommend these based on their overall usefulness.

(Recommending a large number of components may have increased the risk of chips; where not all companies currently make a certain set of fine distinctions, it may not be reasonable to expect all companies to begin making them. Also, a very robust DTD increases the risk that different formatting applications will be able to implement only subsets of the DTD. The group tried to strike an appropriate balance.)

The group also tended to reveal relationships among components by recommending container elements for related subelements, rather than relying on an application to act on them as a group based on their sibling status.

The Language Design paper gives guidelines for turning distinctions into separate elements versus keeping them in one element. Also, the Applications/Purposes paper suggests that similar components should be distinguished by context rather than by having separate unique elements. Where components need different contexts, contents, or attributes, we have recommended different elements, but where a piece of information simply needs to be used differently from another piece that is otherwise similar, we recommend a single element with attributes.

1.3.4 Presentation Independence

According to the Applications/Purposes and Language Design papers, the DTD needs to avoid encoding information that is specific to a particular presentation instance. The latter paper specifies that any necessary presentational components should be attributes, not elements.

To meet this requirement, where possible the design group rejected presentation-related components and relegated the rest to attributes. We also examined components whose traditional name was unnecessarily presentational (for example, ‘‘numbered list’’), and reformulated them in a more abstract way (for example, ‘‘ordered list’’).

1.3.5 Viability of the Document Model

The DTD needs to codify accurate and realistic models for the document types in the scope.

A document type implicitly encodes some assumptions that readers make about how to use documents of that type. The set of assumptions is the document’s model. For example, reference pages are assumed to be able to stand alone and to have certain information contained within them, and as a result, readers can make some predictions about how to use them. For another example, tables of contents are expected in some places and not in others; their usefulness as a navigation tool comes from both their presence and readers’ expectations of where they belong.

Applications are the tools that manifest the entire document model; they make possible certain document features and exclude others. As a result, document model characteristics imply the imposition of certain explicit values on application behavior, such as maximum numbers of division nesting levels. Encoding these values in the DTD ensures that writers can use the DTD in good faith, allowing their intent to be preserved in interchange and in actual use of the information.

The document types in the scope have acquired familiar models over time. The submissions appear to have similar understandings about the models of these types; therefore, our recommendations rarely extend the models found in the submissions.

1.3.6 Component Ordering

The DTD needs to encode consistent, company-neutral orderings of information.

The source ordering of certain groups of components has no functional relation to its order in the presentation instance. Document metainformation contains many examples of this lack of correspondence; for example, ID information is usually used to populate hardcopy title pages, but each interchange partner presents this information in a different arrangement, and each may use different pieces of ID information.

In these cases, each company will need to reorder the information to its own specifications, by using either augmentation software (to create revised SGML source) or normal processing applications (after the SGML source stage). If the DTD allows a random source ordering of these components, every company will need to accept every possible arrangement and change it as required.

The design group has recommended choosing a single consistent ordering of these groups of components, arranged however OSF chooses. This avoids favoring a single company's style of presentation (which also meets the presentation independence principle), while ensuring that the demand on the applications of interchange partners is kept reasonable. Likewise, we have recommended a neutral location for metainformation divisions, such as glossaries, that different companies choose to put in different places in the output.

(As instance-to-instance translation tools such as DSSSL's General Language Transformation Processor (GLTP) become widely available, the problem of reordering content will ease. In the future, this approach might be able to be relaxed. If so, all legacy instances will already conform to the more flexible ordering.)

1.3.7 Planning for DTD Evolution

Given the need for functional DTD additions, such additions need to be planned for and their scope defined carefully.

Over time, a DTD is guaranteed to need functional additions. At any one release, the OSF DTD will naturally be responsible for accommodating known needs of existing information, but it cannot be expected to cover needs that have not yet been articulated. Carefully considering and designing new features at each release allows the DTD to address new requirements successfully and avoids the need for changes to old functionality, which can affect legacy documents adversely.

To adhere to this principle, the design group populated content models sparingly, and recommended components that are known to be needed (because they appeared in at least one submission, or because an imminent need was identified for them). This avoided the creation of artificial semantic distinctions.

The group also designed some ways to protect against changes to applications and legacy documents by recommending ''miscellaneous'' components. These components allow a writer to add semantic distinctions dynamically wherever the set of DTD-prescribed distinctions is insufficient. For example, along with many specific legal notices in metainformation, there is also a miscellaneous notice element with a required description attribute.

Making use of the value of such an attribute must fall to an application; for example, validating applications could ensure adherence to a set of keywords prescribed by an in-house style guide, and processing applications could format information differently based on their assigned keywords. However, OSF could survey these attribute values to assess the need for DTD additions over time, and could even declare in its style guide the keywords that should be used to prepare current documentation for a DTD change in the future. (For example, if a new OSF offering needs a ‘‘graphics command’’ element, the miscellaneous key data element could be used with an attribute value of ‘‘gcmd’’ in the offering’s documentation until the next functional DTD release, when such instances could be automatically converted to use the new graphics command element.[1]) See Section 2.1.6 for more information.

To facilitate high-quality changes over time, the design group recommends that the DTD use parameter entities to provide single points for modification. The information pool subclasses (discussed in Section 2.1.1) are a natural place for the use of parameter entities; these classes reveal a processing and implementation semantic that aids the DTD and tool implementors, rather than a semantic that aids data integrity. Part 11 of ISO Technical Report 9573, *Techniques for SGML Application at the ISO Central Secretariat*, identifies some useful ways to design DTD code for extensibility and maintainability.

1.3.8 Markup Consistency

According to the Language Design paper, similar elements should have content models that are as similar as possible, so that writers can correctly predict how to use those elements. Other features of the markup language, such as attribute availability and ordering, need to be consistent as well.

The design group developed several functional component classes and recommended common characteristics for them, which greatly facilitates markup consistency. For example, the elements in the illustration class contain some common elements. See Section 2.1 for a discussion of these component classes.

1.3.9 Augmentation

According to the Applications/Purposes paper, the DTD needs to allow for augmentation of navigation information, the automatic population of tables of contents and indexes in the SGML source. See Section 2.2.7 for a discussion of our handling of this principle.

1.3.10 Linking for Hypermedia

According to the Applications/Purposes paper and an optional requirement in the RFDTD, the DTD needs to support linking semantics in preparation for hypermedia applications.

Providing specific mechanisms for particular hypermedia applications is outside our scope, but to provide support for such applications, we have recommended all the linking components submitted as well as others that made sense given the markup consistency principle.

[1] This method uses the description attributes as temporary architectural forms; they mark the element’s meaning without using the element’s generic identifier for this purpose.

ISO standard 10744, Hypermedia/Time-Based Structuring Language (HyTime), was discussed in the IAWG as a possible method for implementing links in the OSF DTD, and the related Davenport Group effort proposes specific HyTime-compliant solutions to linking problems in technical publishing. However, it is outside the scope of our work to suggest methods for implementing linking applications. Therefore, the design group has not chosen any of the standard linking mechanisms and their attendant software on the one hand, or a proprietary linking application on the other, for implementing these recommendations in a DTD. Rather, we have recommended semantic components without specifying a way to implement them. (Section 2.1.4 explains processing assumptions for link-related components.)

1.3.11 Localization and Internationalization

The Localization and Internationalization position paper indicates the need for many specific semantic components that would aid translation and localization, computer-aided and otherwise. The recommended components include these.

See Section 1.3.1 for a discussion of a related principle, multilingual document handling, that we determined to be outside our scope.

1.3.12 Relation to the Committee for Common Man

The design group has outlined the overall requirements for reference page interchange in Section 2.2.6.1. We recommend that OSF and CFCM come to agreement on a common SGML encoding for reference pages, as long as these requirements can be met.

The IAWG's policy regarding the Committee for Common Man (CFCM), which plans to standardize man page markup and tools, has been to track that effort and share information and comments. The design group based its original ‘‘reference page’’ work (which also includes reference modules that do not use the `man` tools for their presentation) on early CFCM work, and has shared later revisions and insights with CFCM. Because a CFCM specification has not been published, we cannot claim conformance to a CFCM standard.

1.3.13 Availability of the DTD

To encourage wide use of the OSF DTD in the computer documentation industry and similar communities of interest, it should be made available to these communities.

We recommend that, in addition to making the DTD available with offerings, OSF also register the DTD with the Graphic Communications Association (GCA) and publish it in the upcoming GCA catalog of registered DTDs to allow the general public to review, study, and use the DTD, which greatly increases its value as an interchange tool. (Note that this does not prevent OSF from legally protecting the published DTD's name and contents from misuse or unauthorized change, as it sees fit.)

This document is freely available so that it can be studied and used by outside organizations. Even though this document concludes the formal work of the design group, the SIG needs to maintain it over time to ensure that it continues to reflect SIG recommendations.

1.4 SGML Language Principles

This section describes issues and recommendations surrounding constructs of the SGML language and SGML features. These are only peripherally related to semantic component requirements. Table 1-2 summarizes these issues, which are discussed in the following sections.

Table 1-2: SGML Language Principles

Principle Name	Description
SUBDOC	Avoid use of the SUBDOC feature.
Minimization	Provide a parser that can output normalized instances, and allow OMITTAG minimization.
Including external files	To include non-SGML files, use ENTITY-type attributes and ensure that regular general entities outside attributes are not used.
Declaration subsets	Ensure that local declaration subsets of document type declarations contain only general entity declarations.
Conditional document building	At the OSF-to-supplier stage of interchange, resolve IGNORE and INCLUDE marked sections by filtering material out as appropriate.

Because the use of SGML features and constructs cannot be controlled by a DTD, applications in addition to validating parsers are necessary to ensure the validity of interchanged documents. Some of the following sections recommend the creation of validating applications as adjuncts to a parser.

1.4.1 The SUBDOC Feature

According to the Declarations and Parsers paper, the SUBDOC feature should not be used. This had a direct bearing on our handling of reference pages, for which SUBDOC would have been an appropriate solution; as a result, the design group has recommended a single DTD that encompasses manuals, standalone reference pages, and reference pages inside a manual. Appendix B discusses some implications of our recommendation.

1.4.2 Minimization

An original IAWG requirement was to require interchanged instances to be normalized, that is, to contain no minimization. Recent information has caused the design group to recommend the opposite position. Because files can be normalized easily but reminimized only with difficulty, we recommend that OMITTAG minimization be allowed in interchange, and that receivers take on the burden of normalizing received instances if they do not want to handle minimized files. We also recommend that OSF distribute a parser with its document-processing tools that can output normalized instances. Acquiring such a parser is not expected to be a problem.

1.4.3 Including External Document Files

Using general entities to pull SGML data into the flow of the SGML instance, as is often done to include chapters into a manual, is acceptable. However, where data in non-SGML notation is needed, the design group recommends that the DTD supply an ENTITY-type attribute on an element. Using an entity attribute requires that the entities for the related document files all be declared at the top of the instance file. This aids interchange of all files related to a document.

Using an attribute of some other type for file name information requires an application other than the parser to recognize it as a file, and can obscure which files make up a document.

Using a general entity to pull non-SGML data into the flow of the instance should not be allowed; a validating application should check for illegitimate uses of general entities outside attributes to pull in non-SGML data.

1.4.4 Document Type Declaration Subsets

Internal declaration subsets within the document type declaration in an instance allow local additions and changes to an externally referenced document type. This is essential for the declaration of general entities for referenced SGML and non-SGML files specific to an instance, as recommended in Section 1.4.3. However, other local changes, such as parameter entity, element, and attribute declarations, are very undesirable for interchange. Also, some SGML authoring systems that precompile DTDs cannot yet handle internal declaration subsets of any type.

The restriction with some SGML systems notwithstanding, The design group recommends that instance-specific entity declarations be allowed in interchanged documents, and that other local changes be disallowed. Also, the group recommends that a validating application check for illegitimate uses of the declaration subset.

1.4.5 Conditional Document Building

This document contains recommendations for several specialized components that can be used by applications to customize documents for different purposes. For example, instances of editorial notes can be output in draft documents and hidden in final-form documents. However, a general-purpose mechanism is needed to create document variants covering slightly different subjects from a single set of source files (such as a series of manuals about the same software package on several hardware platforms).

SGML marked sections can serve this purpose, but they are problematic because they can cross element boundaries and therefore disrupt the hierarchy of a document.

The design group recommends that before OSF sends documents to licensees, it resolve IGNORE and INCLUDE marked sections for each variant of an interchanged document by filtering out IGNORE material and removing the marked section markup. (At the first stage of interchange, if OSF is willing to accept marked sections it can negotiate this with suppliers.) A validating application should check for the the presence of marked sections.

Overview of the Recommendations 2

This chapter introduces some conceptual categories for components and gives the design group's recommendations surrounding them; describes our recommendations on issues that cross component boundaries or are of wide interest for other reasons; and summarizes the areas where the design needs further study.

The chapters that follow give the specific recommendations for the main document type organization, the reference page organization, and the contents of the information pool.

2.1 Component Classes

The design group's analysis revealed that the semantic components fall into several classes:

- Section 2.1.1 describes how the document type's organization contrasts with the information pool.
- Section 2.1.2 discusses divisions and components that share some characteristics with them.
- Section 2.1.3 describes information units that function as illustrations.
- Section 2.1.4 describes the classes of linking components.
- Section 2.1.5 describes the technical and general key data classes.
- Section 2.1.6 describes the various descriptor components.
- Section 2.1.7 describes the navigation class of components.

2.1.1 Document Types and Information Pools

A document type has a component organization that makes it unique – a characteristic tree ''shape.'' A computer manual[1] might share some of the same components with a computer-related memo or quick-reference card, but it arranges them in a different way, and its components and shape are very different from those of an unrelated document type such as a tax form.

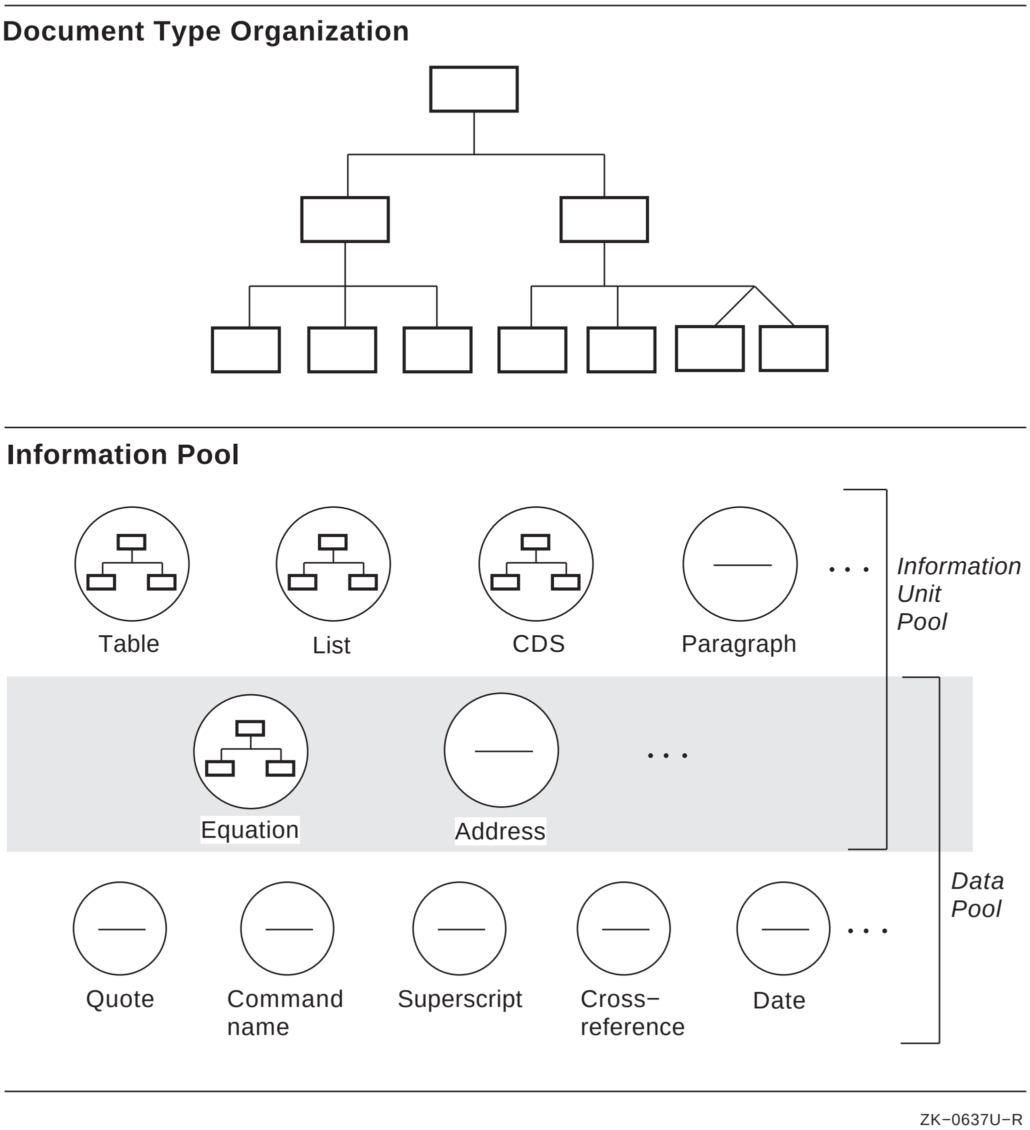
At a low level in most branches of a document type's tree organization, a free mixture of various lower-level elements becomes available. For example, within the document body, a detailed organization of components is prescribed down to the individual division level, but at points within the division elements, a mixture of any number of elements from a certain grouping is allowed. Because a writer must choose among these elements to provide the essential content of the document, they are collectively called the information pool.

[1] In this document, ''manual'' refers to the logical document type, not to the document when it is presented to the user. Thus, the word should not imply a particular presentation form, such as hardcopy or online.

The information pool contains two major subclasses of element mixtures. The information unit (IU) pool offers a general-purpose mixture of high-level elements, most of which have a characteristic internal organization and culminate in low-level elements. The data pool offers PCDATA and a general-purpose mixture of low-level elements, most of which can contain a flat mixture of PCDATA and other low-level elements. (A few kinds of information are represented in both the information unit and the data subclasses.)

Figure 2-1 shows the difference between the document type organization and the information pool and gives examples of elements in the latter class.

Figure 2-1: The Document Type Versus the Information Pool



The design group has recommended a single document type organization that encompasses all the document types in the scope, and a single information pool for use in it. The same information pool would probably be needed inside other document types that were deferred to later phases or considered outside the scope of OSF interchange, such as letters, training materials, and memos. (It is also conceivable, though less likely, that the recommended document type organization could be used with other information pools, such as hardware-only information.) In recognition of the possibility of reuse, we have made clear distinctions between the document type and the information pool in our recommended design.

Different mixtures of the available information units and data are available in various contexts. To meet the evolution principle, the design group developed these mixtures from the bottom up, by looking at each context and determining what should be allowed there. We discovered that only a few different mixtures are needed, which aids the markup consistency principle.

The following sections describe the information unit and data pools in more detail. See Chapter 3 for information on the recommended document type organization for manuals, and Chapter 5 for information on the recommended information pool elements.

2.1.1.1 Information Unit Pool

Table 2-1 describes the contents of each general-purpose information unit mixture.

The numbers were used in the mixture development process (and are used in the tree diagrams), and the names were applied after our analysis revealed their contexts. IU1 is the broadest mixture; it is for use directly in divisions. IU6 eliminates only division-like elements, and is for use inside those elements themselves. IU5 also eliminates bridging titles; it is used inside the IUs other than the division-like elements, where bridging titles are not appropriate. IU2, IU3, and IU4 are used in contexts that (such as metainformation and ‘‘text alternatives’’) where minimal descriptive ability and formatting capability is needed or desired.

The work of determining information unit mixtures revealed various subclasses shown in Table 2-1, such as the note class and the dynamic illustration class. The IUs in these subclasses should be available or unavailable as a group in IU mixtures because of their similarity, which meets the markup consistency principle. Other sections in Section 2.1 describe some of these classes, and Chapter 5 is organized along the lines of the table for convenience.

IU mixtures appear in the tree diagrams in this document with an occurrence indicator of either ‘‘zero or more’’ or ‘‘one or more.’’ Except for the special occurrence rules for bridging titles (noted in Section 5.1.15), IUs are assumed to be allowed in a free mixture.

Table 2-1: General-Purpose Information Unit Mixtures

	Division Content (IU1)	Division-Like Content (IU6)	IU Content (IU5)[a]	Simple (IU2)	Simple with Titles (IU3)	Simple with Address (IU4)
Paragraph[b]	X	X	X	X	X	X
Generalized lists:[b] Simple Ordered Unordered Definition list Segmented item list	X	X	X	X	X	X
Specialized definition lists: Argument list Return code list Message list Bibliography list Glossary list	X	X	X			
Notes: Note Caution Warning Danger Editorial note Note collection	X	X	X			
Address block	X	X	X			X
Misc lines display	X	X	X			
Excerpt	X	X	X			
Models: Routine model Command model File model Data structure model	X	X	X			
Table	X	X	X			
Figure	X	X	X			
Computer dialog/state	X	X	X			
Equation	X	X	X			
Dynamic illustrations: Sound clip Video clip Application launch	X	X	X			
Division-like elements: Supplementary note Procedure Scenario	X					
Bridging title[c]	X	X			X	

Table notes:

a. Certain individual IUs exclude themselves or their subclasses from this mixture. Chapter 5 discusses these cases. Problems frequently occur in the nesting of IUs within other IUs (and data within other data); testing will need to uncover any such problems that are not already addressed in our recommendations.

b. Paragraphs and generalized lists can normally directly or indirectly contain other IUs not allowed inside IU2, IU3, and IU4. The IU mixture at lower levels must conform to the most restrictive mixture in which it appears; in other words, any IUs not allowed in these mixtures must be disallowed inside paragraphs and generalized lists used as part of these mixtures. For example, the disclaimer element in metainformation contains an IU3 mixture; therefore, a disclaimer can contain a paragraph, but the paragraph cannot contain a figure.

c. Bridging titles have IUs as peers. However, they are not available in a completely free mixture with other IUs. See Section 5.1.15 for more information.

2.1.1.2 Data Pool

Table 2-2 describes the contents of each general-purpose data mixture.

The data1 mixture is the broadest; it is used in general data contexts where real subject matter content is supplied, such as paragraphs and key data elements that encompass whole regions of data (such as quotes). The data2 mixture is the narrowest; it is used for alternate titles and in some simple metainformation contexts, so that no sophisticated formatting is required. The data3 and data4 mixtures are mostly used in key data elements that are intended to contain a ‘‘word’’ of material (such as telephone numbers). The data5 mixture is mostly used in titles; it eliminates cross-references (to avoid infinite regression when titles are used as generated cross-reference text) and graphic data.

Table 2-2: General-Purpose Data Mixtures

	data1	**data2**	**data3**	**data4**	**data5**
PCDATA	X	X	X	X	X
Equation data	X				X
Graphic data	X			X	
General key data	X	X	X	X	X
Technical key data	X		X	X	X
Cross-reference links	X				
Footnote links	X			X	
Other data links	X				X

Data mixtures are indicated in the tree diagrams in this document with an equal sign (=) and the name of the mixture. The intended meaning is that the individual elements and PCDATA are to be in an ‘‘or’’ grouping with the occurrence indicator ‘‘zero or more.’’

See Section 2.1.4 for a description of the linking component classes mentioned in this table, and Section 2.1.5 for a description of the key data classes.

2.1.2 The Division Class and Related Elements

The document type organization is made primarily of divisions, which are cohesive, labeled collections of information that can be arranged in a hierarchy that gives them semantic context. A division can be one of three types:

- Generalized

  A generalized division, such as a subdivision of a chapter, can be used to hold any kind of information of the writer’s choosing.

- Specialized

  A specialized division, such as a bibliography, can be used to hold only a particular kind of information.

- Miscellaneous

  A miscellaneous division, such as a subsection in a preface, can hold any kind of information, but its contents are required to be identified as if it were a specialized division.

Specialized and miscellaneous divisions exist to allow readers and applications to predict their contents with varying levels of accuracy and thus manipulate them in sophisticated ways. More predictions can be made about specialized divisions that can occur only once than about those that can occur multiple times in different places. For example, if a document bibliography is present, it can be predicted to contain all the relevant bibliography information for the manual as a whole, but because multiple message divisions can appear in various places, no single one of them can be predicted to contain information about all the messages relevant to the document’s subject matter.

Just as for information unit and data mixtures, the design group developed the content models for divisions from the bottom up, by looking at each context and determining what should be allowed there. Table 2-3 is a summary of the content models of all the divisions that are generally available in the body and supplementary information.

Table 2-3: Content Models for Divisions

	Navig	Summary	Biblio	Title/alt title	IU-level content	D[*n*+1]	Ref/Msg D
Superpart				X		X	
Part	Same as doc level	X	X	X	IU1	X	X
D1/intro D1	Just TOC	X	X	X	IU1	X	X
D2				X	IU1	X	X
D3				X	IU1	X	X

Table 2-3: (continued)

	Navig	Summary	Biblio	Title/alt title	IU-level content	D[*n*+1]	Ref/Msg D
D4				X	IU1	X	X
D5				X	IU1	X	
D6				X	IU1	X	
D7				X	IU1		
Ref D				X	Ref pages	Ref subdiv	
Msg D				X	IU1, msg lists	Msg subdiv	

The navigation, summary, and bibliography components are available on the higher division levels, just as for the document as a whole. Generalized divisions at level 5 and below do not contain the specialized divisions, because the specialized versions have their own subdivisions two levels down. See Section 2.2.5 for more background on our recommendations for divisions, and Section 3.4 for text descriptions and tree diagrams describing the recommendations.

Division-like elements and bridging titles share some features of divisions without being true divisions.

Division-Like Elements

The scenario, procedure, and supplementary note IUs form the division-like element class. They have some characteristics of divisions in that they are all are cohesive, labeled collections of information, but they are arranged somewhere inside a division in a series of IUs, rather than as part of a hierarchy of similar division-like elements. Like true divisions, they can contain bridging titles.

Bridging Titles

A bridging title is a label for a location within some division or division-like element; its purpose is to assist the reader's navigation of the discussion once the division has been located. It is similar to a division title without actually labeling a true division.

The model for technical computer documents assumes that, to a greater or lesser degree, a division can stand apart from its context and still make some kind of sense. This is why, for example, typical hypertext applications treat divisions as distinct topics or articles.

However, there is sometimes a need within a division to break up the discussion further into pieces that the reader can easily understand. The pieces cannot be pulled out of the larger context of the whole division, but distinctions between the pieces can help the reader absorb and retain the material. Bridging titles label the points at which the discussion changes gear; they provide organizational structure without providing additional hierarchical structure, allowing the division in which they occur to remain intact.

Even though bridging titles are available at the IU level, with IUs as peers, because of their special relation to divisions they cannot be used in a free mixture with the other IUs. See Section 5.1.15 for more information.

2.1.3 The Illustration Class

Some information units are, by their nature, sensitive to the way they are presented to the user. For example, while a list can encode information abstractly and be presented a number of different ways, a table must be presented with certain appearance information intact. (This distinction was first noted in the RFDTD.) We called these IUs illustrations.

The IUs fall in different places along the continuum of presentation sensitivity, but the design group identified tables, figures, computer dialogs/states, equations, sound clips, video clips, application launch, and possibly models as illustrations.

Illustrations have certain features in common, such as needing particular kinds of descriptor information. We identified the possible illustration features and examined each IU to determine which features it needs. Table 2-4 shows our recommendations for illustration features.

Table 2-4: Common Illustration Features

	Table	Figure	CDS	Equation	Dynamic illus's	Models
Title	X	X	X	X	X	
Caption	X	X	X	X	X	
Alternatives	X	X		X	X	
Annotations		X	X			X
Key	X	X				
Alternate notations	X		X	X		

Titles, captions, and alternatives are discussed in Section 2.1.6.

Annotations are attached to the illustration they describe rather than being part of the flow of text, to ensure that they are associated with their illustration. Points in an illustration can link to individual annotations. (Annotations in an illustration are in an annotation collection element, rather than an ‘‘annotation list’’; each annotation has a stronger logical connection to its referenced location than to the other annotations within the illustration.)

A key is different from an annotation in that it defines pictorial or text-based conventions used in the illustration, rather than explaining particular features of the subject matter.

An illustration can contain information encoded in a way that is typically associated with another component. For example, figures contain graphics expressed in some kind of graphic notation. However, tables may also need to represent their content with such a notation, which nonetheless does not make them figures. Therefore, an illustration is a semantic component, whereas its content is only a physical representation. The illustrations noted as having alternate notations offer one or more encodings other than the usual one.

The concept of illustrations played little part in our recommendations of allowable mixtures of information units; we used it only in populating the content models of these particular units themselves.

2.1.4 The Link Classes

The design group's analysis revealed four basic classes of links. Writers make certain assumptions about the behavior of some of these links; these document model features result in expectations about processing and formatting. The processing assumptions are noted for each class of link.

- Cross-reference links

  A cross-reference link produces a textual suggestion to the reader to seek out the other end of the link. It appears at the data level. It can point to locations and elements in the current document and other documents. A processing application is expected to generate text that describes the thing being pointed to, which the writer is responsible for incorporating into the flow of data (for example, working it into a sentence).

  This kind of link adds intelligence to the document; writers choose to add them where appropriate.

- Footnote links

  This kind of link attaches explanatory text to a particular location in data. It requires an application to generate text, a number, or some other symbol outside the flow of the component in which it occurs, allowing the reader to access the explanatory text. There are no requirements on how the footnote is to be accessed; printed books typically put footnotes at the bottom of the page and table footnotes at the bottom of the table. Footnotes are similar to illustration annotations, which are often rendered in ''callout lists,'' or as hotspots in the illustration that the reader can click on to see the annotation.

  Whereas footnote links are available in general-purpose data mixtures, annotation links are available only in computer dialog/states and command and file models. (Annotations without the links are available in figures.)

  This kind of link adds intelligence to the document; writers (or readers) choose to add them where appropriate, and must write the explanatory text.

- Other links from the data level

  This kind of link simply associates two or more pieces of information. For example, a command name in text can point to the command's reference page. Some of the links in this class, like this example, appear on specific key data; others are available in general-purpose data mixtures. An application could provide various results for such links, such as a hotspot, icon, or other text or symbol that is outside the flow of text. However, no special processing is expected or required; the link is merely an association.

  This kind of link may or may not add intelligence to the document; many of these links can be augmented or otherwise generated, rather than being supplied by writers.

- Navigation information links

  This kind of link points from somewhere in the augmented navigation information (a TOC or index) to the appropriate element or location in the document.

TOC links do not add intelligence; some index links can be generated, but many of them must be added by someone with knowledge of the document and the subject matter.

See Section 2.2.7 for more information on recommendations for navigation information.

2.1.5 The Key Data Classes

General and technical key data are subclasses of the data-level elements.

Key data elements identify small pieces of information that readers or applications may want to ‘‘key on,’’ such as a command name. For applications, this may mean building automatic index entries for occurrences of a command name in text, or linking occurrences of a command name to its reference page. For readers, this typically means giving special typographic emphasis to the command name.

Technical key data elements mark software-related information. General key data elements mark information that is general-purpose in nature, such as acronyms.

2.1.6 The Descriptor Class

Many elements have features that exist for the sole purpose of describing or labeling the element to various degrees of precision. Following are explanations of the various descriptors we recommended:

- IDs

  Many elements, such as divisions and some IUs, have identification (ID) attributes on them. These attributes are optional; they are for use only when an instance of a particular element may need to be referred to from a particular place elsewhere.

  IDs can be added in other places, as long as they are not required attributes. However, for elements that appear only once in an entire document, as with most metainformation, we did not recommend an ID because the element could already be uniquely identified.

- Titles and alternate titles

  Many divisions, division-like information units, and illustrations have titles to label their contents. The title is assumed to be for reader consumption. Titles contain a data5 mixture, which allows all links from data except cross-references. Cross-references are not included because they generate text and can cause an recursive-processing burden when titles are used as the generated text for cross-references.

  Alternate titles are available on divisions that have titles, to allow for processing or display deficiencies. They are similar to the ‘‘alternatives’’ described below; this is why they contain a data2 mixture, which is simpler than the regular title’s mixture.

- Captions

  Captions on illustrations optionally provide an extended description in addition to a title. Like titles, they contain a data5 mixture.

- Alternatives

  Some illustrations need an alternative in case of processing or display deficiencies or reader problems in perceiving certain kinds of information (for example, deaf readers when presented with a sound clip). We have recommended the optional inclusion of text and/or graphical alternatives to allow writers to supply information for these cases. Text alternatives contain one or more elements from the simple IU2 mixture, to make sure they can be handled by all display devices.

  Text alternatives can function as a kind of abstract for reader consumption, or they can be a container for whatever description the sender thinks the receiver needs to reconstruct or replace the problematic material. For example, for an audio clip, the sender could describe what the clip was intended to accomplish as well as describing the actual audio recording. For text-based illustrations, the alternative could include any raw text that was supplied in bitmap or other inaccessible form (such as figure callout text, or the text that appears in a table) so that a receiver can reconstruct a facsimile of the intended illustration if necessary.

  Section 2.2.6.2 discusses alternatives as they relate to reference page display restrictions.

- Abstracts and summaries

  A manual can have an abstract to describe its contents more briefly than the title does, and a summary to describe its contents in more detail than the abstract does. Some levels of division also have summaries. The different gradations of description could be used in various kinds of documentation catalogs.

- Descriptive keywords

  A manual can optionally have keywords for retrieval functions, and a reference page can optionally have keywords for the building of a permuted index.

- Description and topic attributes

  Miscellaneous elements always have a attribute for a description of the contents. These miscellaneous components are intended to allow the DTD's semantic capabilities to expand as needed; therefore, these attributes are required.

  Description attributes let writers supply descriptive strings of their own choosing (although applications can enforce any mandated keywords). Topic attributes require writers to choose from among a standard list of tokens.

  Where a token list is provided, it may be a good idea to include a ‘‘miscellaneous’’ or ‘‘other’’ token and to add a description attribute to the same element. This would allow even a hardcoded list of tokens to be extensible in a way that does not compromise the interchange data protocol.

2.1.7 The Navigation Class

For convenience in discussing augmented tables of contents and indexes, the design group has applied the collective name ‘‘navigation information’’ to them. This should not imply that other methods of document navigation are not possible; for example, cross-references and introductory text both provide methods of navigation to a reader. However, these latter kinds involve material in the flow of text, and as such are not good candidates for augmentation.

See Section 2.2.7 for more information about this class.

2.2 Issues of General Interest

This section describes our recommendations for issues that cross component or class lines, or that deal with common DTD design or computer documentation themes. Following are the topics covered:

- How IUs fit into paragraphs
- Information that can appear at both the data and the IU levels
- The relation of specialized divisions to their contents
- Guidelines for title occurrence
- Distinct versus nested semantic components for divisions
- Reference page issues
- Augmented navigation issues
- Guidelines for whether IUs should be required in divisions
- Making a documents division hierarchy correspond to that of another document
- Localization handling

2.2.1 IU Content in Paragraphs

A common DTD design issue is whether paragraphs can contain IUs, or whether they can contain IUs only as peers.

A paragraph is traditionally (if presentationally) understood as an unbroken block of narrative text that wraps freely. In our understanding of the information pool, this maps to paragraphs containing only data. However, in computer documentation, paragraphs are often used to introduce and explain lists, computer dialog/states, and other IUs that are typically set off from the flow of data, and the same train of thought may continue after the end of the displayed IU. In these cases, the paragraph can be interpreted to contain the other IU, even though this seems to contradict the traditional definition.

(The concept of displayed IUs is somewhat presentational. Even though only the illustration IUs can be argued to need ‘‘setting-off’’ treatment, typically other IUs receive this treatment as well. For example, lists could be incorporated into the data flow of a paragraph with their item markers, but this is rarely, if ever, done in computer documentation. Therefore, the document model directs us to assume that all IUs other than paragraphs will break the current paragraph flow and be set off from the previous material.)

Based on the way paragraphs are used in documents in the scope, the design group concluded that paragraphs can legitimately contain other IUs. Therefore, we recommend that paragraphs be allowed to contain all elements from the IU5 mixture except the following:

- Paragraphs themselves

  Direct nesting of paragraphs within themselves is foreign to the model of the documents in the scope (although it may make sense in certain military and legal documents).

- Dynamic illustrations

  These illustrations are perceived in a way ‘‘bigger’’ than a paragraph is; because they are dynamic, they must be experienced over time. Therefore, for a paragraph to contain them seems inappropriate.

- Notes

  Notes are like specialized paragraphs (or collections of them), and are therefore peers or parents to paragraphs, not children of them.

- In some contexts, all IUs other than generalized lists

  In the IU2, IU3, and IU4 contexts, IUs other than generalized lists are disallowed because of the restrictive nature of these contexts.

  Prototyping revealed that in IU2, IU3, and IU4 contexts, an SGML exclusion was necessary for disallowing these other IUs, which is an awkward way to achieve the desired result. Nonetheless, we recommend having restricted content models in the places where the IU2, IU3, and IU4 mixtures are indicated, as well as allowing those other IUs inside paragraphs and lists in normal contexts.

- Specialized definition lists

  Originally, our recommendations allowed paragraphs to contain specialized definition lists. Prototyping revealed that in specialized divisions that require a certain IU to be present to satisfy the expected content of the division (for example, message lists in message divisions), writers would have had to ensure that at least one message list was present directly inside the division rather than inside a paragraph. Because this presents usability problems for writers, specialized definition lists should be disallowed from paragraphs.

This recommendation results in a paragraph being the only IU that can have a free mixture of data and IUs within it.

Since displayed IUs can be both peers to paragraphs and children of them, writers can decide to nest any particular IU within a paragraph based on whether the paragraphs seems to contain the IU logically.

Illustrations are the only IUs inside paragraphs that have titles. We have treated each kind of illustration, whether titled or untitled, as a single semantic component; therefore, titled illustrations as well as untitled ones can appear in paragraphs. Typically, presentation choices for IUs are not made depending on their titling and their relationship to paragraphs, but they could be. For example, a titled IU inside a paragraph could be interpreted as being anchored to that place in the flow of text, whereas one outside a paragraph could be interpreted as being outside the flow (and possibly be allowed to ‘‘float’’ to a different location).

2.2.2 Information in Both IU and Data Pools

In contrast to the displayed IUs discussed in Section 2.2.1, data elements are expected to be incorporated into the flow of character data. (This is often called ‘‘inline’’ data.) Several kinds of data are assumed to be able to appear either inline or displayed, at the writer’s option.

We concluded that writers legitimately have control over this behavior; it serves as a kind of emphasis of the data’s importance, and we wanted to be able to retain the writer’s intent in interchange. Therefore, each kind of information needs two components to represent it:

- Equation information can be encoded either as equation data or as an equation illustration.
- Address information can be encoded either as an address data or an address block.
- A quotation can be encoded either as a data-level quote or an excerpt. (Note that these two components have other characteristics that further distinguish them.)
- A graphic can be encoded either as graphic data or within a figure IU. In other words, when a graphic is displayed, it is known as a figure. (Several illustration contexts other than figures also contain graphics.)

Since the data and IU versions can both appear inside a paragraph but the IU version can appear as a paragraph's peer as well, these pairs need to be elements, not attributes on a single element. In some of the cases, two elements are needed in any event because the two versions have different characteristics (content models or attributes).

2.2.3 Specialized Divisions and Their Contents

Bibliography, glossary, reference, and message divisions all control their contents similarly.

Requiring the Presence of the Specialized Contents

The specialized divisions require the presence of at least one instance of their respective specialized contents. This ensures that applications can effectively use the information found in it, and discourages TAS by making it more difficult to use such a division for its presentational effects.

This entails eliminating their respective specialized contents from any introductory text, because in all cases except that of reference divisions, the specialized contents are IUs.

Requiring the presence of the specialized contents helps ensure that the specialized divisions will be used for their intended purposes. However, prototyping revealed that the special content models required for this recommendation may complicate the management of information pool entities and may contribute to some greatly increased capacities and quantities for the DTD. If basic restrictions on using the SGML language make this arrangement impossible, an application besides the parser may be needed to ensure the presence of the specialized contents.

See Section 2.2.1 for a discussion of another issue related to this recommendation.

No Mixing of Specialized Contents and Subdivisions

Inside any one instance of a division, ‘‘real content’’ cannot be mixed with further levels of division. For example, a message division cannot directly contain both a message list and further message subdivisions.[2] If this mixing were allowed, it would effectively allow ‘‘super-descriptions’’ of messages. For example, if a message division were to contain a message list and two subdivisions one level down, it would imply that the subdivisions contain information pertaining to the messages found at the top level. The top-level message list effectively ‘‘contains’’ the divisions. We felt this was an inappropriate use of divisions to convey message

[2] Different subdivision branches, however, can contain message lists at different levels.

structure.

In cases where such hierarchy of information is appropriate, the IU itself is allowed to contain further nested IUs of the same type. Thus, for example, messages can have submessages, so a message entry description is allowed to contain message lists. However, glossary entries cannot contain glossary lists and reference pages cannot contain reference subpages because the document model assumes these kinds of information to be flat and modular in nature.

2.2.4 Appearance Guidelines for Titles

The design group applied the following general guidelines to the recommendations for titles on other elements.

Regular generalized divisions have required titles on them. The title serves as a label for the contents of the division, and is required because no way exists to predict the contents of the division.

Specialized divisions that can appear multiple times, such as message divisions, also have required titles; although the type of contents is known, the specific contents of a particular instance of a message division cannot be predicted.

Specialized divisions that can appear only once (for example, the description and return code divisions in reference pages) do not contain titles. Their contents are wholly predictable, and individual writers are typically not intended to have control over their labeling; it is a global style consideration. If titles are disallowed here, the applications of different receivers can easily and consistently generate the title they prefer for each division (for example, See Also, References, or Related Information).

However, note that the document bibliography and glossary in metainformation, and the specialized introductory divisions, do have required titles. This is to provide flexibility within a company, say at the project level, to name these divisions; we are assuming that a style sheet that normally supplies mandated title text would be appropriate only at the company-wide level. This flexibility is probably needed in the introductory material, but if it is not desired for the metainformation divisions, titles can be disallowed here.

Miscellaneous divisions that complement some set of specialized divisions (for example, miscellaneous subdivisions in the preface) contain required titles, but they also have a description or topic attribute that allows the contents to be identified. Receivers may be able to control in-house attribute use in order to generate globally standard titles, discarding the supplied title, but this method is imperfect.

An optional title is recommended in a few places where its containing element behaves sometimes like a division and sometimes like a simple collection of information, or where it was considered important to have a default title string generated whenever none was supplied.

Among the IUs, only the division-like elements, notes, and illustrations can have titles. The document model generally assumes that titled illustrations will somehow be numbered, and probably collected in a special table of contents; the same may also be true of the division-like elements. However, in certain contexts, such as in the introductory material or nested inside another IU, numbering and related treatment may be inappropriate. Allowing the titles in all contexts meets the data integrity and markup consistency principles; rather than disallowing titles in certain contexts, we recommend that the desired formatting rules be identified so that applications can behave appropriately.

2.2.5 Division Components

Generalized divisions and subdivisions make up the main content of the manuals in the scope. The design group identified the following document model features of these divisions:

- The highest levels of division are somewhat similar to the manual as a whole; they function more as an organizing mechanism than as a direct container of essential IU-level material.

  For example, just like manuals, superparts and parts (and sometimes even chapters and appendixes) divide up the subject matter by general, fairly imprecise criteria such as ‘‘user information’’ versus ‘‘system administrator information.’’ Some of the higher levels of division can have some features of manuals, such as tables of contents and possibly even indexes. (Table 2-3 in Section 2.1.2 shows this trend.)

- The highest levels of division each have a somewhat distinct role in a manual, whereas the lowest levels have less distinction between them.

  For example, the IU-level material that appears directly in a part is constructed specifically to serve a part-like function (that is, as an introduction to a part), and IU-level information directly in chapters and appendixes is similarly specific. Promoting or demoting the information in the division hierarchy is very likely to require rewriting. Some of the companies represented in the design group have performed experiments attempting, for example, to move level-3 material to level 1; the experiments showed that rewriting was essential. The fact of the higher-level distinctions explains why writers can start a particular manual at a superpart, part, or chapter level.

  By contrast, below the chapter or appendix level, promotion and demotion have less effect on the nature of the IU-level material; a second-level division functions very much like a third-level division. After a certain point, no useful distinction can be made. The point at which the distinctions fade away is somewhat relative, depending on the level at which a particular manual’s divisions start.

- Divisions, especially but not exclusively higher-level ones, are expected to be read by themselves, without necessarily having other surrounding and containing divisions present to provide context.

  Because typical computer manuals are used as problem-solving tools, they are constructed so that a reader can focus in on the relevant information without first needing to read surrounding and containing material.

  Reference pages, many hypertexts, and manuals that use structured writing techniques all reinforce this feature of the document model by mandating that divisions have a very limited number of levels, providing context by cross-referring to other material rather than by including it or relying on being adjacent to it. Traditional manuals do have more levels of division than these document types, but they are nonetheless accessed as if each division can stand alone.

- There is a limit to how many total division levels can be used.

  This limit appears to reflect a fundamental boundary on how much any subject matter can be subdivided and still make sense; essentially, this is a usability requirement. The design group reexamined the issue of level limits at its June 1992 meeting, and concluded that our nine recommended levels, including superparts, were sufficient to cover all needs of documents in the scope.

  (Note that the 20 levels that some U.S. Department of Defense documents have actually span both the document type and the information levels; at least half the levels function as titled paragraphs. The DoD document model is unrelated to those in our scope.)

Two different approaches were considered for divisions: distinct division components for each level (named divisions) and a single division component that nests within itself in some manner (generic divisions). The design group analyzed these approaches in light of the document model features.

Named divisions provide a limit on division levels, and if they are separate elements as opposed to attributes on a single element, they also accommodate the need for each level's contents to be controlled separately. Generic divisions can be used to collapse lower-level divisions that are identical in function into a single component; however, generic divisions cannot control the limit on the number of levels. Overall, named divisions have many more advantages than generic divisions in encoding an accurate document model for interchange.

Therefore, we recommend that named division elements be used, with a maximum of nine levels: superparts, parts, and first-level divisions through seventh-level divisions.

Our recommendation for the specialized divisions is based on the same reasoning, except that fewer levels of division were found necessary for them: bibliography and glossary divisions have a maximum of one additional level, and message and reference divisions have a maximum of two additional levels.

Note that bridging titles, described in Section 2.1.2, greatly extend a writer's capacity to structure information; inside any division in a manual, bridging titles can be used to provide organizational structure without providing additional hierarchical structure.

The approaches to divisions also have differing effects on two other areas: reuse of information in multiple documents and draft manual development. In both areas, named divisions have some overall advantages.

Reusability Concerns

To make efficient use of developed information, organizations often attempt to reuse information in several documents simultaneously. Often the SGML general entity mechanism is used for reusing small amounts of IU-level material, such as tables, in multiple documents; however, reusing an entire division subtree in this way can be problematic.

The Applications/Purposes paper specifies that where possible, context rather than unique elements should distinguish between components, so that information can be easily moved and reused. If generic divisions are used, divisions can theoretically be reused at several different levels simultaneously, even if they have internal subdivisions; this is not the case with named divisions.

However, as noted in the discussion of document models, promoting and demoting division subtrees at the higher levels often causes a need for rewriting. The same is true of attempting to use a high-level division subtree at multiple levels simultaneously, except that the rewriting problem cannot be resolved; the material can be optimized for only one context at a time.

In fact, experience suggests that reusing any division subtree that has its own internal hierarchical organization is troublesome; a division that is designed to have a certain organization in one context is likely to be completely inappropriate for other far-removed contexts. Much more likely to succeed is reusing a single topic with no internal hierarchical organization: a flat or modular description. Even if named divisions are used, this can be accomplished with the use of general entities to hold the IU-level content of divisions rather than the divisions themselves. If additional organization is needed inside the modules, bridging titles are an appropriate tool for the task because they keep the module intact.

Draft Manual Development Concerns

DTDs used in draft manual development are not fully in the scope of this effort; OSF will be shipping static snapshots of documents to licensees, and suppliers engaged in an active document development effort with OSF are likely to negotiate modifications to the interchange data protocol. However, the design group made the following observations that might affect DTD variants that are closely related to the interchange DTD.

A common situation in editing a draft document is the need to promote or demote a whole subtree of divisions. As the discussion of document models points out, at some high levels this cannot be accomplished in an automatic way; some text must almost always be rewritten. However, at the lower levels, rewriting may not be necessary. Generic divisions have the advantage of making such promotion or demotion a fairly simple procedure: the addition or deletion of one or more surrounding division elements.

However, other consequences arise from using generic divisions. Writers must explicitly indicate the closing of such a division with an end tag; in structured editing environments this is not a problem, but in unstructured environments the burden of tracking end tags (which can sometimes be very far away from each other in a file) can be immense. In addition, in many structured environments built-in editing features ease the promotion and demotion of subtrees. Therefore, if a DTD variant with generic divisions is shared among writers in both types of environments, it can inconvenience those in the unstructured environment while providing little benefit to those in the structured one.

2.2.6 Reference Page Issues

After receiving much helpful OSF and SIG input on a draft of our reference page recommendations, the design group reconsidered issues surrounding reference page document types and display restrictions at our June 1992 meeting.

The following sections discuss our revised recommendations.

2.2.6.1 Reference Pages as a Document Type

A reference page document type can be highly specialized, as if it were a database record with a field for each topic (the database approach); highly generalized, as if it were a regular manual where a writer has total control over the organization and contents (the manual approach); or somewhere in between.

Current markup methods encode reference pages using the manual approach, with methods outside the markup language enforcing any desired document model constraints. For example, the Description section is required in every reference page; this is clearly a feature of the document model, but current markup methods cannot express or enforce it. Instead, the Description section’s presence, labeling, and location must be enforced through templates, style guides, and editorial review. This situation is problematic for licensees in the following ways:

- The validation of a reference page as conforming to features of the document model cannot be determined easily.
- Presentational style standards that are decided on a per-company basis (such as the ordering and labeling of reference page components) can be enforced only through the laborious modification of the actual source instance.
- Receivers have difficulty accurately predicting where information resides in order to move, remove, or otherwise manipulate it.

These problems stem from markup that has too little data integrity and too much dependence on presentational features. Reference pages conform to a highly organized document model that has many natural semantic distinctions; if these features are encoded with a sufficient degree of data integrity and presentation independence, reference pages can finally become valuable resources both for the extraction and building of documents (including entirely new document types, such as quick-reference guides, command summaries, and so on) and for online information retrieval.

Our draft recommendations advocated a middle approach that offered a fairly high degree of semantic precision. However, its operative model assumed that the writer would have control (through the source instance) of quite a bit of final-form presentation ordering, enough to undermine the usability of interchanged reference pages. Of additional concern, these presentational features did not correspond with OSF's current presentational style standards.

Therefore, the design group recommends that a database approach be used for the reference page document type. Specifically, we recommend that each well-defined subject area (such as a Description section) have an individual element, and that the subject areas all have a consistent, company-neutral ordering.

Some semantic flexibility is naturally needed to accommodate subject areas that do not have standard ‘‘fields’’; therefore, we recommend using miscellaneous elements with description attributes to meet these needs. Reference pages encoded this way provide enough semantic and syntactic predictability to meet the interchange, data integrity, presentation independence, and component ordering principles.

Note that we have also recommended that the OSF and Committee for Common Man (CFCM) reference page document types correspond if possible; see Section 1.3.12 for more information.

Chapter 4 details the specific recommendations for the document type organization for reference pages.

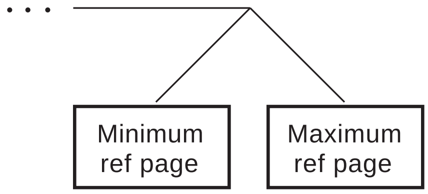
2.2.6.2 Reference Page Display Restrictions

In the following discussion, it is assumed that because the source instances will not be in `man` macro form, applications in addition to, or in place of, the `man` tools will need to operate on SGML reference pages. The OSF *DTD Documentation Tools Plan*, available to OSF members, details this issue more thoroughly.

Current `man` reference-page formatting and display tools and their target display devices have some problematic restrictions. Many illustration-type elements (such as figures, dynamic illustrations, and some tables, as well as the graphic data element) are not well suited to being displayed in character-cell environments, and because of their presentation sensitivity, using text alternatives in place of the original information often distorts the intention of the writer.

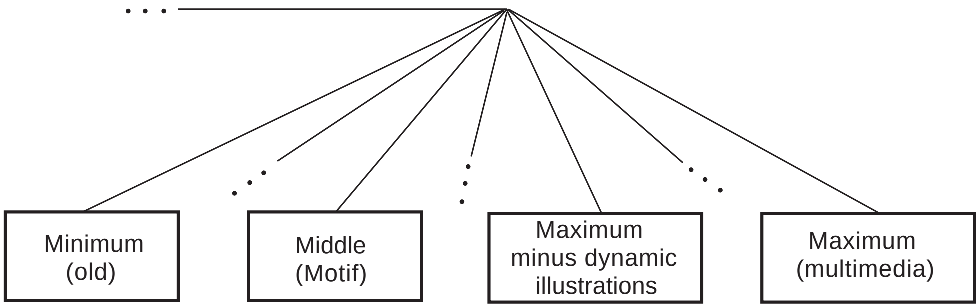
Because of the problems of receiving such illustrations in interchange with no way to process or use them successfully, the design group originally recommended to exclude the problematic IUs from reference pages. However, this would have resulted in a document type that encodes presentation restrictions in a way harmful to data integrity, because the excluded kinds of information were nonetheless legitimate information for a reference page. For example, in a Motif reference page, a screen result encoded as a graphic might need to be supplied; the exclusion would improperly prevent it from being in the page.

However, receivers of reference pages need some practical way of managing reference pages so that the restricted form can be produced, since the `man` mechanism is still the only way most people access reference pages. One alternative is to have two different kinds of reference page elements, one with the restrictions and one without, so that an interchanged reference page has some of way of controlling its contents and warning a receiver what it contains:



ZK−0634U−R

However, this can lead the document type design down a slippery slope, resulting in an encoding of display restrictions far beyond what would be reasonable under the presentation independence principle:



ZK−0635U−R

LP We concluded that the DTD is an inappropriate place to control the presence of problematic illustrations; applications should handle this job. However, an interchanged reference page needs a way to indicate that it contains none of the problematic illustrations, so that receivers can predict what applications and/or display devices will be required for any one page. Therefore, we recommend that a reference page have an attribute that indicates the type of contents, either basic or enhanced.

The exact contents that are considered basic versus enhanced need to be identified; over time, some enhanced contents will presumably turn into basic contents as new applications are developed.

For any enhanced reference page, text alternatives to illustrations become especially important. While we have not recommended that text alternatives be required on these elements, obviously their presence and their quality will be essential for receivers that need to reconstruct or replace them. For illustrations that have any text in them, such as tables, or figures that contain callouts, text alternatives could be used to provide the raw text that the illustration contains. Receivers could use this to construct suitable facsimiles.

2.2.7 Augmented Navigation Information Issues

OSF and SIG input on a draft of our recommendations uncovered some issues around interchanging augmented tables of contents and indexes.

Originally, the understanding was that such information would let receivers avoid having to perform a great deal of processing on received instances, so that hardcopy formatting applications could be simpler and online hypertext access mechanisms could operate directly on the source form of the document.

However, for the information to assist applications enough to warrant its presence, it would need to contain presentational information in its most harmful form. For example, to assist hardcopy formatting, a table of divisions would have to contain the page numbers that the divisions start on. As soon as the document was modified in the slightest way, this page number information would be invalidated and would need to be regenerated. Even for online hypertext applications, any document modifications could invalidate the augmented information, causing it to need regeneration (or causing it to be used in incorrect form).

In addition, the only way for a receiver to validate the augmented information received in interchange is to perform a process on the source that is similar or identical to the one that generated the information in the first place, negating the advantage of having received it.

Finally, unless extremely precise semantics of tables of contents and indexes are agreed to by all the interchange partners, there is no guarantee that the information generated by a sender will meet the needs of a receiver. For example, a sender may feel it is appropriate to list divisions down to the third level in a table of divisions, whereas different senders may prefer two, four, or six levels, or may prefer to put both brief and annotated versions of a table of contents in their documents. Such decisions are in the domain of presentational style; organizations should be able to control them by methods outside the SGML source.

Therefore, the design group has withdrawn its specific recommendations for the contents of a navigation information container.

This is not to suggest that augmenting this information has no merit. However, a more precise motivation for having and using it needs to be identified, so that the resulting design can meet the needs of information creators and interchange partners.

This reasoning applies as well to the special table of contents for reference divisions that was discussed with OSF and the SIG and to augmented permuted indexes; see Section 3.4.4 and Section 4.4 for more information.

Our recommendations retain a navigation element in metainformation with free-form character data content, and we hope that further research will contribute to structuring its contents as appropriate over time. Section 3.2.4 describes our original set of recommendations.

2.2.8 Guidelines for ‘‘Empty’’ Divisions

The design group considered whether to require at least one IU to be present in the body of divisions.

The goal of doing this would be to require proper introductory text before a division is subdivided or to provide a true text body within the division, as is generally considered good style. However, even if we required an IU to be present, writers could always circumvent the requirement by using empty paragraph tags. In any case, a parser cannot check for high-quality introductions or division bodies, and we do expect humans, along with parsers, to examine documents that are being interchanged to determine their quality.

Therefore, we recommend that only divisions at ‘‘terminal’’ nodes (such as D7 and message subsubdivisions, where the IUs present are guaranteed not to be introductory, but by definition make up the body of the division) have at least one IU present. We recommend this with the expectation that a writer who has a motivation for using these very low-level divisions already has text in mind for them.

2.2.9 Making Division Hierarchies of Two Documents Correspond

Documents that describe conformance to a standard typically need to mimic the standard document’s structure for the purpose of explaining conformance to specific items. For example, current POSIX conformance documents typically reproduce the hierarchical structure (chapters and sections) of the original POSIX (1003.1) standard, reproducing the whole structure, large parts of it, or only the exactparts that are relevant.

Today, simulating the numbering and titles of the original divisions seems to be sufficient, but in the future, linking to the other document may be practical. The design group has not made a strong recommendation in this area. However, we made the following observations.

Depending on the style chosen by an organization, different parts of the structure are revealed. For example, some organizations print all the division numbers and titles, but supply text only in divisions where there is something to say about conformance (usually starting to print at the first ‘‘non-empty’’ division, however). Others print only the non-empty divisions.

Accommodating these different styles with one solution is difficult. An attribute on divisions that initializes their sequence numbers is the simplest (so that each organization can control what divisions they supply and what their titles and numbers are). However, it may invite Tag Abuse Syndrome if writers use them to control numbering during draft development and the numbers are incorrect when the final document is interchanged.

Adding an attribute in metainformation that marks the document as a conformance-type document would enable applications to use the right numbering scheme. For example, if the document is conformance-type, the attributes could be used in formatting. If it is not and has a draft status of final, the attributes could be ignored.

Alternatively, conformance attributes on each division could indicate the section number and title of the original document to which the division corresponds.

This area needs further study.

2.2.10 Localization Handling

One requirement was to be able to mark information as needing localization attention in cases where no specialized component is available. For example, addresses have certain localization needs, and address components can be found in instances and dealt with; however, there is no component specifically for culture-specific jokes.

Three different localization statuses must be available: do not localize, localize with caution (in other words, do not translate automatically), and localize at all costs because there is a legal requirement to do so. The latter two can apply to the same set of information: localize at all costs, but do so with caution. In addition, comments may need to be attached to each instance of a localization problem.

The design group recommends that these be encoded as a **localization status** attribute with a token value indicating which of the three situations applies, and a **localization comment** attribute explaining the circumstances. (The latter could be an SGML comment in the source, but putting necessary information in comments is problematic because some representations of SGML do not recognize comments.)

We do not recommend putting these attributes on all elements; the following text-based elements are sufficient:

- Divisions
- Scenarios

  Often, scenarios must be customized to a particular culture to be effective.
- Paragraphs

  This allows whole paragraphs to be marked.
- Phrases

  This allows a phrase within data (such as inside a paragraph) to be marked precisely.

2.3 Areas Requiring Further Study

Following is a summary of areas in these recommendations that are incompletely specified or that have known problems, along with the sections where they are discussed. Areas where we have simply mentioned possible alternative designs or suggested future enhancements are not mentioned here, but are noted in each specific recommendation in the text.

Note that this document is not intended to supplant rigorous testing against real documents. Therefore, even the recommendations that present a fairly complete picture of the components and their arrangement may need to be rearranged or enhanced to accommodate existing needs.

- Making standards conformance documents correspond to the original standard specification document (discussed in Section 2.2.9)
- Complex table encoding (discussed in Section 5.1.9)
- Equation encoding (discussed in Section 5.1.12)
- Notations (discussed in Section 6.2)
- Return code lists (discussed in Section 5.1.3.2)
- Branching lists (discussed in Section 5.1.2.3)
- Conditional document building (discussed in Section 6.1)
- Reference pages and their relation to eventual CFCM specifications (discussed in Section 1.3.12)
- Reference page metainformation (discussed in Section 4.1)

- Augmented navigation information (discussed in Section 2.2.7 and Section 3.2.4)
- Models, particularly the routine and data structure models (discussed in Section 5.1.8)
- Data mixtures (discussed in Table 2-2)
- Data-level components in general (discussed in Section 5.2)
- The generalized link component (discussed in Section 5.2.7)

The design group is available during the DTD implementation period to answer questions on the recommendations and provide input on new design decisions. Issues should be mailed to the `sig-doc-di@osf.org` distribution list.

See Appendix C for information on how the prototype DTD implemented these incomplete recommendations.

Manual Organization Recommendations 3

This chapter describes the recommendations for the organization of the manual document type. In this chapter, only the allowable contexts for reference pages are discussed; see Chapter 4 for recommendations for the content model of reference pages.

The tree diagrams in this chapter (whose notation is explained in Figure A-1) provide a useful overview of the recommendations, but the recommendations are explained completely only in the text. Where no indication of element occurrence is given in the text (for example, ‘‘zero or more’’), the element is assumed to be required. Also, attributes are less completely specified than elements; their names, types, possible values, and defaults are typically implied by the accompanying explanation.

See Table 2-1 and Table 2-2 for an explanation of the IU and data mixtures found in various contexts in the manual organization.

3.1 Top-Level Organization

The tree diagram in Figure 3-1 summarizes the top-level document organization.

Because the scope includes both standalone reference pages and computer manuals that can contain collections of reference pages, and because of the requirement to avoid use of the SUBDOC feature, we recommend that a single DTD be used to accommodate both document types. Appendix B discusses some implications of this.

The top-level element corresponds to a single logical document. Section 3.2 contains recommendations for information on volumes (fractions of a single document that exist because of presentational constraints); ‘‘volumes’’ of the type that consist of a whole document and make up an *n*-volume set are considered simply individual documents in these recommendations.

The DTD (and the top-level element) should have a name descriptive of the document type. For example, ‘‘osfdoc’’ is not descriptive enough; something like ‘‘techmanual’’ would distinguish it from other document types that might eventually be created.

The top-level element can contain either a reference page or a series of high-level containers that corresponds to a manual. For a manual, the top-level element contains the following, in order (to meet the component ordering principle):

- **Metainformation**, which describes characteristics of the manual itself. It contains control information such as the draft status descriptor, as well as descriptive text information such as legal notices. This component is required because certain portions of its content must always be present during interchange.
- Optionally, **introductory information**, which contains any material that introduces the document and its subject matter to the reader. It can be said to contain both descriptive information about the document (metainformation) and information on the document’s subject matter, and so has been made distinct from the preceding and following components. It primarily contains a preface.

- A document **body**, which contains the core subject matter material. It primarily contains ‘‘chapter’’ divisions.
- Optionally, **supplementary information**, which contains subject matter that is isolated from the core content because it is nonessential or is a summary or quick reference of core material. It primarily contains ‘‘appendix’’ divisions.

These structures correspond only vaguely to the traditional (but presentation-oriented) front matter/body/back matter construction of printed books. They are invisible collectors for related information, to be used for localized content model control and information retrieval.

3.2 Metainformation

The tree diagram in Figure 3-1 summarizes the metainformation organization.

Metainformation contains two basic kinds of information. Its small descriptions of document characteristics This information is typically used by applications for tasks such as document storage and management, building hardcopy front matter, and providing online access based on retrieval descriptors. Some of this control information is likely to be processed without accessing the entire document; therefore, it has a highly prescribed structure and simplified information content.

Its descriptive divisions, which are similar to real document content, allow information content that is identical to that in the rest of the manual.

Metainformation first contains the following groupings of small descriptions, in order:

- **Document ID** information, which contains numbers and histories associated with the document. It is required because some of its contents are required to be present for interchange.
- Optionally, **product ID** information, which describes characteristics of the computer systems that the document applies to.
- **Notices**, which contains legal, safety, and other notices intended for the reader’s consumption. It is required because some of its contents are required to be present for interchange.

We grouped this information into three containers, but it could be grouped differently as desired. Providing such subgroupings eases the management of the various small pieces of information.

Within each grouping, to meet the component ordering principle we recommend that there be a fixed order of OSF’s choosing. Here, we have ordered the elements roughly by logical function.

We recognized that different organizations make use of different subsets of these fine distinctions. This is an area where chips are likely to occur in interchanged documents, but we felt that the usefulness of having the distinctions available in the interchange DTD outweighed this risk.

Metainformation then optionally contains the following descriptive divisions, in order:

- **Document navigation** information, which provides document entry points to the reader to facilitate finding relevant content. This is optional because it can be generated or augmented at a later stage.

- A **summary** division, which describes the document more fully than the abstract.
- A **bibliography** division, which lists source information for other related documents.
- A **glossary** division, which defines terms used in the document.

Section 1.3.6 explains why these divisions, which might be located at various places in different presentation instances, are gathered here under metainformation.

Metainformation also has two attributes that describe any printed volumes into which the document might have been divided. The volume information is important as a record of what the document creator did, but is inherently presentational; therefore, it must be encoded in attributes. The **number of volumes** attribute records the number of volumes into which the document was divided. The **identification references** attribute points to a series of IDs of the locations or elements (presumably high-level divisions) that form the breaking points for the printed book.

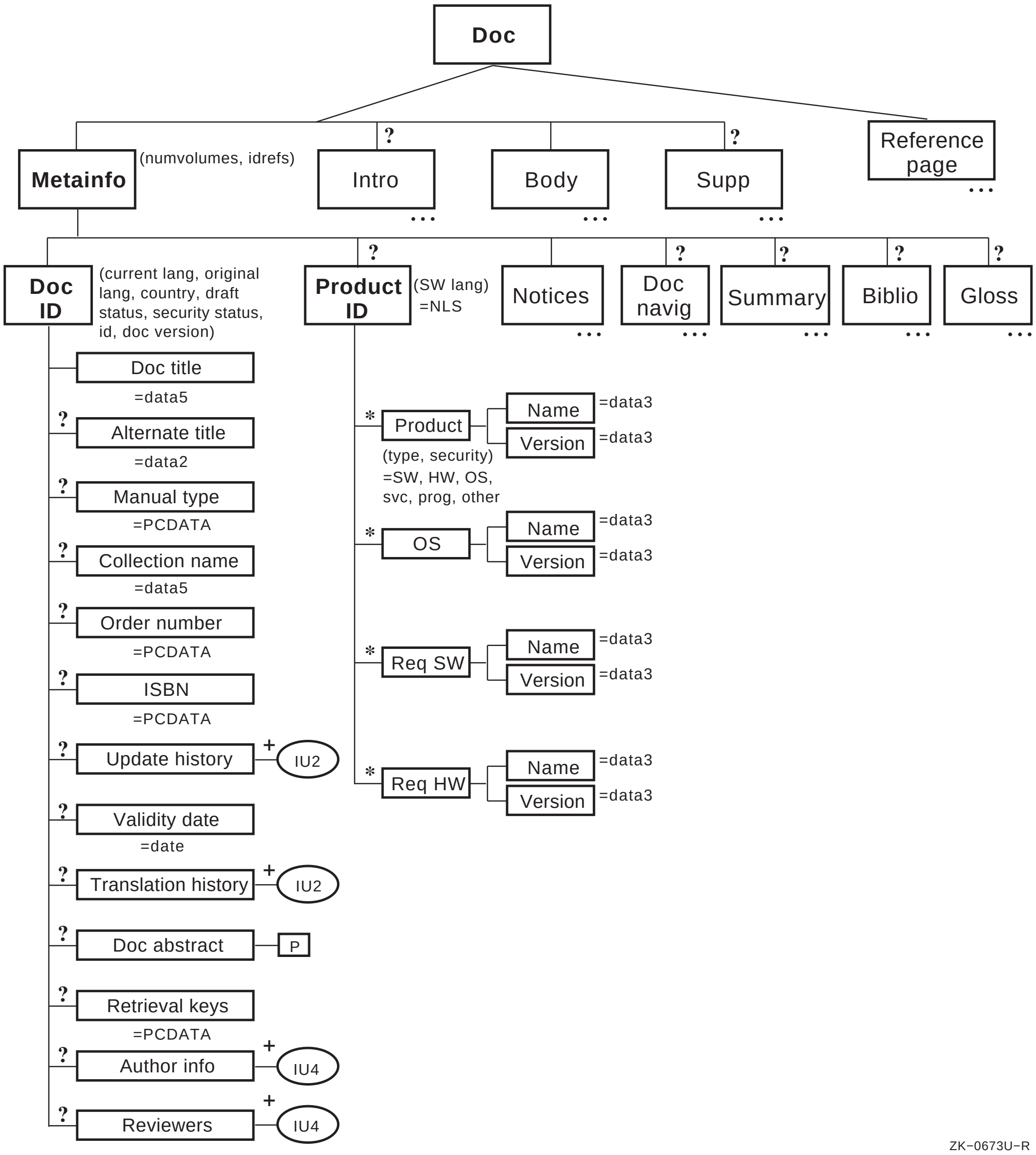
In several places within metainformation where little descriptive capacity is needed or desired, the simple IU mixtures (IU2, IU3, and IU4) are available. In the metainformation divisions, however, the normal range of IUs is available.

In some places where we have recommended using attributes, the components could be elements instead. (Some of the attribute-like information in document and product ID information is already in elements because it needs to contain key data.)

3.2.1 Document ID Information

The tree diagram in Figure 3-1 shows the document ID information.

Figure 3-1: Metainformation: Document and Product ID



The document ID information contains the following series of elements, in order. All of these elements except the document title are optional.

- The **document title**, which is the primary label for the document. It contains a data5 mixture. The title is required because it is the most basic form of document identification.
- The **document alternate title**, which is a title for use when constraints of the display device or the processing preclude using the regular title. It can be thought of as an alternative to the title, like the text alternative to a figure. Therefore, it has more restricted content: data2.

- The **manual type**, which is a label for the document audience or purpose (and could be assembled as part of the document title on output). By convention, we expect that its content will be a choice from among a list of standardized entities, for example, ‘‘Maintenance Manual,’’ ‘‘Administrator’s Guide,’’ or ‘‘Reference Manual.’’ (See Section 6.3 for more about recommended entity sets.) It contains PCDATA.
- The name of the document **collection** or library to which the document belongs (which could be assembled as part of the document title on output). It contains a data5 mixture, to match the document title.
- The **order number**, or part number, which is the number a customer would use to order the document. It contains PCDATA.
- The **ISBN** (International Standard Book Number). It contains PCDATA.
- The **update history**, which describes the document’s revision history, its relation to previous versions, and so on. It would typically contain a series of release dates. It contains one or more elements from an IU2 mixture, to ensure that it is not empty if present. IU2 allows segmented item lists, which might be appropriate for date histories.
- The **document validity date**, which is the date the document enters final form, ready for use. This element is different from the date element in key data, but has the same attributes and content model. See the discussion of the date element in Section 5.2.3 for more information.
- The **translation history**, which describes the series of translations performed on the document over its lifetime. It contains one or more elements from an IU2 mixture, to ensure that it is not empty if present. IU2 allows allows segmented item lists, which might be appropriate for date histories.
- The **document abstract**, which is a concise description of the document. It contains a single paragraph (described in Section 5.1.1). Note that paragraphs can normally contain other IUs; only generalized lists should be allowed inside it here.

  See also Section 3.2.5 for information on the document summary.
- **Retrieval keys** information, which contains PCDATA and consists of keywords descriptive of the document’s content (for example, MOTIF or INSTALLING). This could be a series of attribute values on the element rather than being element content; it might be printed with the document as well as being used as search values, so either solution is appropriate.
- **Author information**, which is intended to contain the name of each author of the document, how to contact them, and their sponsoring organizations. It contains one or more elements from an IU4 mixture, to ensure that it is not empty if present. (The key data element for document author information is expected to be used if authors’ names are supplied.) IU4 allows the address element, which can used for authors’ addresses.
- Information on **reviewers** of the draft document. It contains one or more elements from an IU4 mixture, to ensure that it is not empty if present. IU4 allows the address element, which can be used for reviewers’ addresses.

Document ID has several attributes:

- The **current language** attribute indicates the language in which the document in its present form is written.

- The **original language** attribute indicates the language in which the document was first written.
- The **country** attribute indicates the country for which the document was first written.
- The **draft status** attribute indicates the stage of development the document is in, such as whether it is at working draft, snapshot, or final form stage. Different snapshot substages might also be indicated. Its setting could be used to control, for example, whether draft distribution restriction warnings appear in the presentation instance. A token list of attribute values should be supplied.
- The **security status** attribute indicates the distribution restriction category of the document, such as OSF-members-only and open. (For an interchange DTD, a status of company- or OSF-confidential is inappropriate.) A token list of attribute values should be supplied.
- The **identification** attribute uniquely identifies the document for use by other documents when they point to this document or to elements or locations inside it.
- The **document version** attribute indicates the revision level of the document. For example, ‘‘B’’ or ‘‘02’’ might be the version for a document that is in its second revision. The version is a required attribute; even though the same information might appear in the order number, organizations use different schemes for representing the version in the order number, and this separate component ensures that the version can be located in an interchanged document.

3.2.2 Product ID Information

The tree diagram in Figure 3-1 shows the product ID information.

The product ID information supplies name and version information for the computer-related product being documented. For each element inside product ID, a **product name** and **product version**, in that order, are required. They each contain a data3 mixture.

Product ID contains the following series of description elements, in order; individually none of this information is required to be supplied, but if the product ID element is present, it cannot be empty.

- One or more pieces of **product** information, each of which identifies the product being documented.

  Each product description element can have two attributes. The **product type** attribute can have a token value indicating the nature of the product: at least software, hardware, operating system, service, program, and other. The **security level** attribute can have a token value indicating the level of U.S. Government security functionality to which the product conforms, such as B1 and C2. (This attribute is unrelated to the security status attribute on the document ID element.) It should have a token list of attribute values.
- One or more pieces of **operating system** information, each of which identifies (if the product is not itself an operating system) an operating system on which the product can run.
- One or more pieces of **required software** information, each of which identifies software required to be installed for the product to be operational.

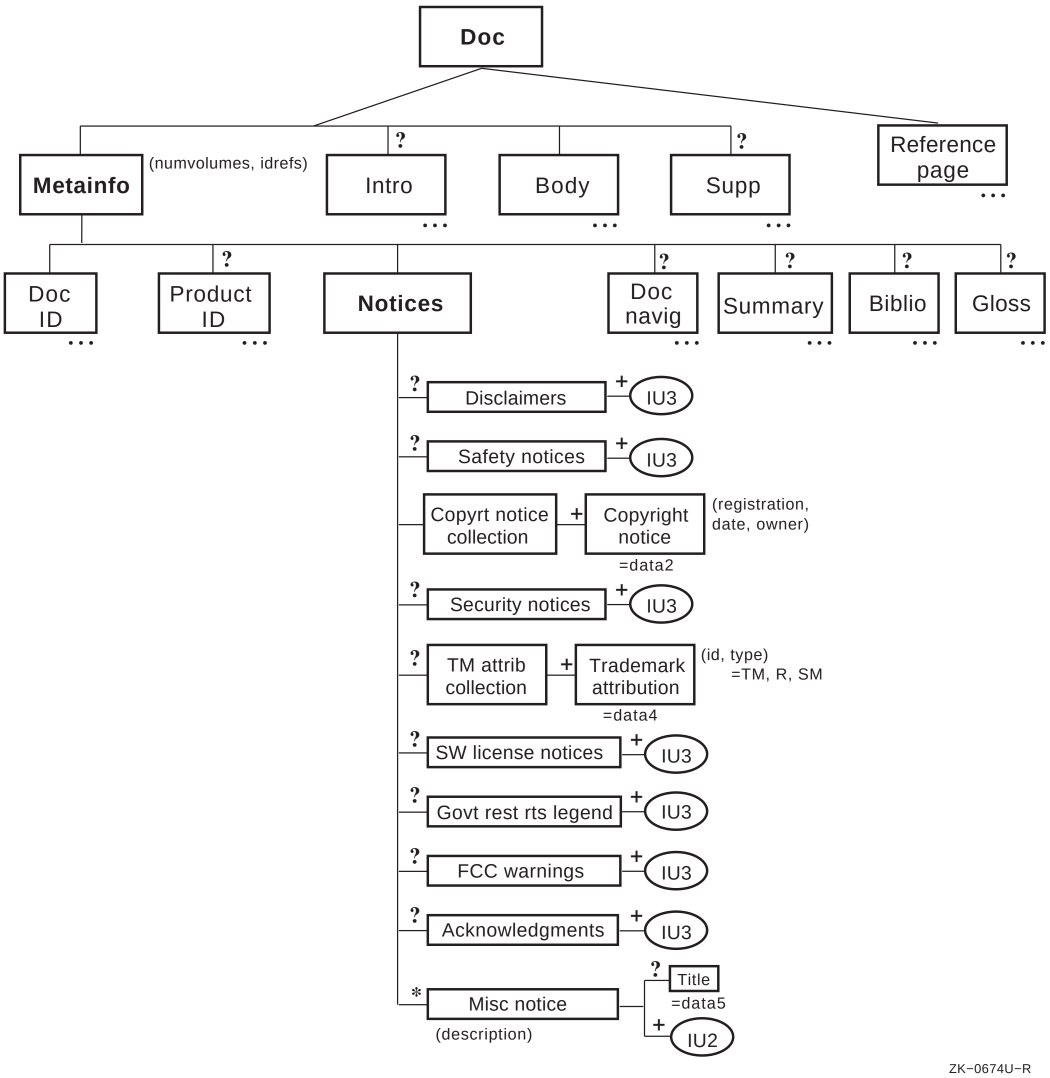
- One or more pieces of **required hardware** information, each of which identifies hardware required to be installed for the product to be operational.

Product ID has an attribute for the language of text that appears in the software; it should have a token list of values corresponding to national language support (NLS) codes. (See the Glossary for more information.) This is useful to know if message translations are being supplied in message entries; see Section 5.1.3.3 for more information.

3.2.3 Notices

The tree diagram in Figure 3-2 shows the notices information.

Figure 3-2: Metainformation: Notices



The notices element contains a series of notices, in the following order; only the copyright notice collection is required. Most of the notices contain one or more elements from an IU3 mixture, to allow bridging titles; notices with a different content model are explained below.

- A **disclaimers** element describing the creator's or distributor's responsibility for performance of, and errors in, the software and the document.
- A **safety notices** element describing any (typically hardware-related) safety considerations.
- A required **copyright notice collection** containing one or more **copyright notices**, which are legally required notices of copyright ownership and conditions that can contain a data2 mixture. Any dates are expected to be marked up with the date key data element. A copyright notice has an optional attribute for the **copyright registration** number and required **copyright date** and **copyright owner** attributes (to ensure that this information is supplied and can be manipulated). This results in date and owner information being supplied in both the attributes and the content, but the arrangement of the various pieces of the notice needs to remain in the control of the information creator.
- A **security notices** element describing the security and distribution restrictions for the document. This is different from the document ID security status attribute in that it contains text meant for reader consumption and cannot actually control aspects of processing or distribution.
- A **trademark attribution collection** containing one or more **trademark attributions**, which are legally required information on the ownership of trademarked terms used in the document. An attribution can contain a data4 mixture. Graphics are allowed in the data because trademarks often consist of special graphics. Each attribution can have an **identification** attribute for reference from trademarked terms mentioned in text, and a **trademark type** attribute to indicate the type of trademark: TM (regular trademark), R (registered trademark), or SM (service mark).
- A **software license notices** element describing the license needed to use the documented product.
- A **government restricted rights legend**.
- An **FCC warnings** element describing radio emissions from the product.
- A **acknowledgments** element, which details legally required information on the people or organizations who had some involvement in creating the documented product.
- Zero or more **miscellaneous notices**, each of which has a **description** attribute to label its contents. A notice contains an optional **title** (different from the document title element) and one or more elements from an IU2 mixture. The title is optional so that, if titles are generated on output for the specialized notices, labels can likewise be supplied for the miscellaneous notices. This notice has no bridging titles allowed in it; individual notices that need setting off should be in individual miscellaneous notice containers, so they can be labeled. If a certain topic that has special content needs is used repeatedly over time, a later version of the DTD can accommodate it.

  This title element is a different component from the document title. It contains a data5 mixture.

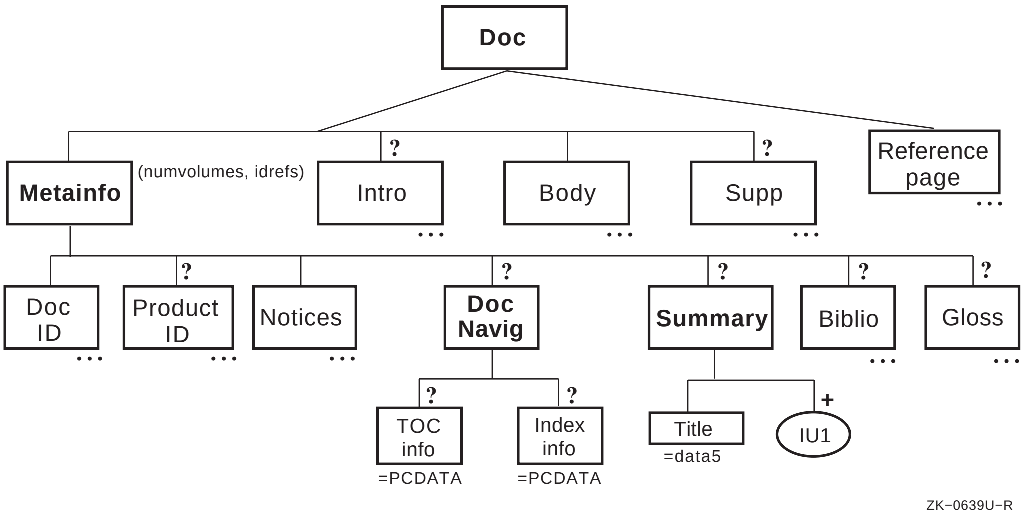
The specialized notice components are distinct elements because each should be able to appear only once, and because such information might be suitable for retrieval. The miscellaneous notice is for extensibility if additional distinct notices need to be supplied.

Many of the notices are described here with a plural component name. This means that any relevant notices are collected in a single element, separated by paragraphs if desired. The exception is copyright notices and trademark attributions, which may need to be operated on individually as well as in a collection.

3.2.4 Document Navigation Information

The tree diagram in Figure 3-3 shows the augmented navigation information.

Figure 3-3: Metainformation: Navigation and Summary



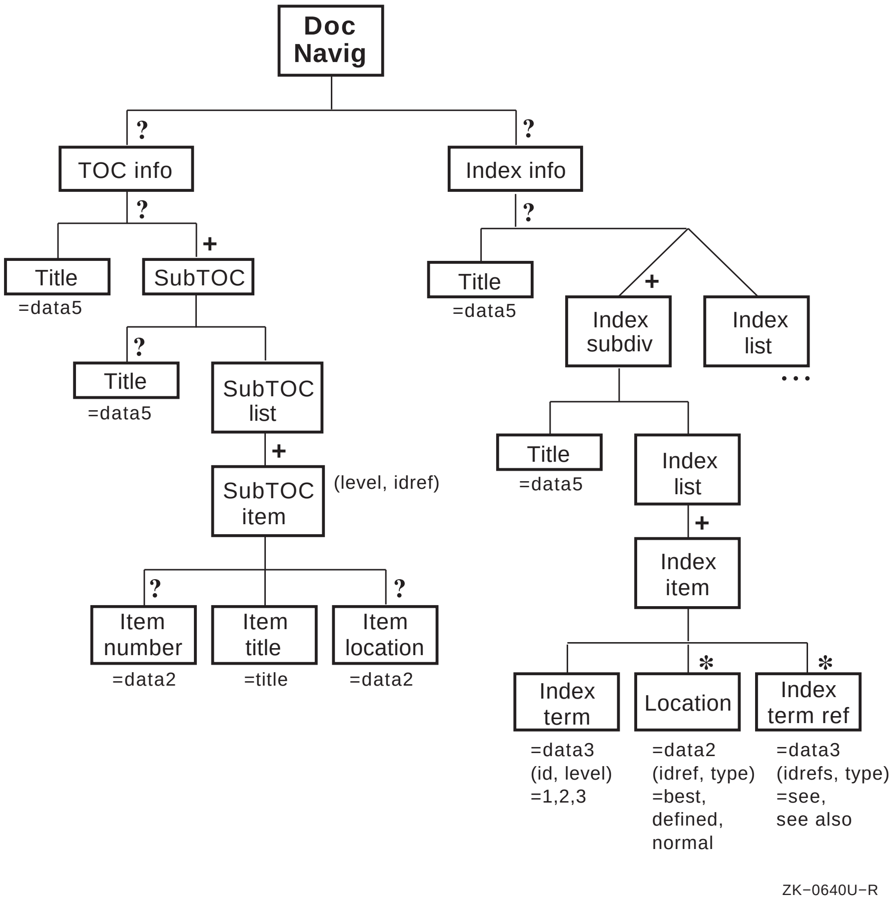
The document navigation information is supplied to allow augmentation. It contains the following, in order:

- An optional **table of contents information** (TOC) element that contains PCDATA
- An optional **index information** element that contains PCDATA

If the navigation information is present, it cannot be empty.

The PCDATA is essentially a placeholder until further study in this area can be done; Section 2.2.7 discusses the design group's recommendations for further work on augmented navigation information. Figure 3-4 shows our original recommendations for the organization of augmented navigation information; these recommendations have been withdrawn, as Section 2.2.7 explains, but the tree diagram and its explanation are offered here to contribute to future research in this area. Note that the presentation dependence of the location components is of particular concern.

Figure 3-4: Possible Document Navigation Organization



The TOC and index are formulated as divisions; they contain titles and can have subdivisions.

3.2.4.1 Table of Contents (Informational)

The table of contents collects listings of titles or other labels for document elements, and helps the reader locate the elements. It contains a **title** and one or more individual subtables of contents (**subTOCs**). In traditional books, the first part of the TOC lists divisions; for our purposes, this is considered a subTOC within the main collector, which can be followed by other subTOCs (such as a list of figures).

Each subTOC contains an optional **title** and a **subTOC list**. The subTOC title is optional to reflect the fact that the listing of divisions usually has no individual title in printed output, and to enable this option. (The subTOCs do not have an attribute labeling the type of element being listed inside them.)

The subTOC list can accommodate any kind of listing, as long as the the target elements have a title (or at least allow some sort of label to be constructed for them) and, typically, an ID. SubTOCs could be created for divisions, illustrations, division-like elements, and any other titled element.

A subTOC list contains one or more **subTOC items**, each with a **subTOC level** attribute for the nesting of the element it describes (for example, a division nested inside another division, the only usual case). Since each subTOC item corresponds to an element in the document, having an attribute for the level, rather than nesting the items, is appropriate; the series of subTOC items itself is flat. Validating the supplied attribute values would be problematic if the TOC were manually created, but we are assuming the level can be correctly generated by an application. The subTOC item also has an **identification reference** attribute that links the item to the element in the document.

A subTOC item contains the following, in order:

- An optional **item number**, which is the numbered label the element has in the document, for example, ‘‘Section 1.4.3’’ or ‘‘Figure 5–9’’. It contains a data2 mixture.
- A required **item title**. It is different from titles that label elements; it merely records the title by which the item, which appears elsewhere, is labeled. It contains the same data mixture as a title does, which is data5.
- An optional **item location** element content that describes the location where the element can be found. Typically the content would be a page number, which is bound to a single presentation instance by its nature. It contains a data2 mixture.

3.2.4.2 Index (Informational)

The index collects sorted listings of labels attached to document locations, and helps the reader get to the locations. It contains, in order, a **title** and either an **index list** or one or more further **index subdivisions**. In traditional books, each subdivision would be a sorted subcollection of labels, such as all the labels that start with the letter ‘‘A’’, but an application can group the labels however is desired.

Each index subdivision contains a **title** and its own **index list**. An index list contains one or more **index items**.

An index item contains one or more of the following:

- A **index term** containing a data3 mixture, with an **identification** attribute for cross-references from other index items and an **index term level** attribute to indicate the nesting level of the term. The level attribute should accommodate a maximum of three levels. Having an attribute for the level, rather than nesting the terms, is convenient. Validating the supplied attribute values would be problematic if the index were manually created, but we assume the level can be correctly generated by an application.

  It is expected that there would be only level-1 term in each item, and that it would appear first. For example:

  bananas
  in cake
  in hair-care products

  The **bananas** term in this entry is at level 1, and the other terms are at level 2.

- Zero or more pieces of **location** information containing a data2 mixture (the same as the subTOC item location), each of which has an **identification reference** attribute that links the term to the document location where the index entry resides, and an **index reference type** attribute that has a token value indicating the kind of location reference: at least best reference, location where the term is defined, and normal reference.

The content of the location information describes the location in the document. Typically it would be a page number, which is bound to a single presentation instance by its nature.

- Zero or more **index term references** containing a data3 mixture, each of which has one or more **identification reference** attribute values indicating the index items being referred to.

  The content of the index term reference describes the referred-to index term. Typically it would be some text such as ‘‘See’’ or ‘‘See also’’ and the series of terms that identifies the portion of the relevant index item.

  It has an **index term reference type** attribute to indicate the type of reference: at least see and see also.

Note that an index item is different from an index entry in the text; the correspondence is not one-to-one. Rather, each occurrence of a location element in an index item corresponds roughly to a single entry in the text.

3.2.5 Document Summary

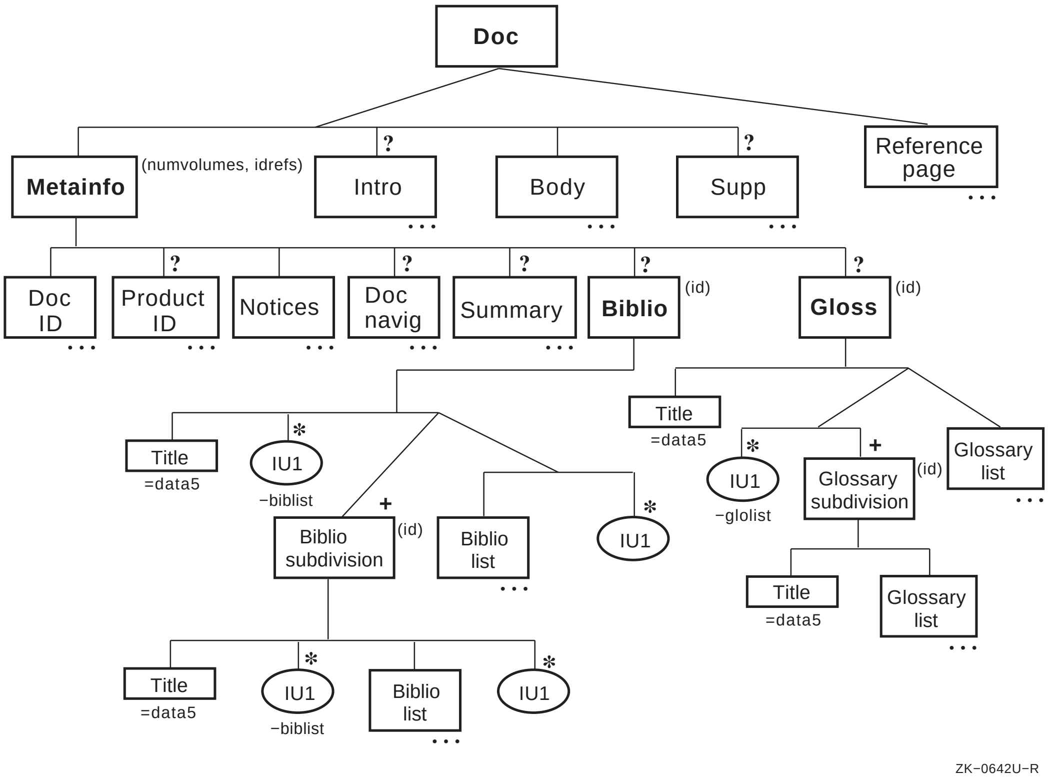
The tree diagram in Figure 3-3 shows the document summary.

The document summary division is an extended description of the document; it could be considered to contain all the information in the division summaries (although merely concatenating the summaries, if they existed, would likely produce a poorly written document summary). It is a flat division with no subdivisions inside it; it contains a **title** and one or more elements from an IU1 mixture. IU1 is needed to give flexibility in allowing the summary to contain brief versions of ‘‘real’’ subject matter.

3.2.6 Bibliography

The tree diagram in Figure 3-5 shows the bibliography.

Figure 3-5: Metainformation: Bibliography and Glossary



The bibliography division is a collector for source information on related documents. It contains the following, in order:

- A **title**
- Zero or more elements from an IU1 mixture that excludes bibliography lists (see Section 2.2.1 and Section 2.2.3 for information on some consequences of this)
- Either one or more **bibliography subdivisions**, or a required **bibliography list** followed by zero or more elements from an IU1 mixture (which can include additional bibliography lists).

A bibliography subdivision contains the following, in order:

- A **title**
- Zero or more elements from an IU1 mixture that excludes bibliography lists (see Section 2.2.1 and Section 2.2.3 for information on some consequences of this)
- A required **bibliography list** followed by zero or more elements from an IU5 mixture (which can include additional bibliography lists)

Bibliographies are very likely to be manually created, rather than created through augmentation. Therefore, the content model ensures the presence of at least one bibliography list, while allowing introductions and explanations between lists.

A bibliography has one optional level of subdivision; for example, internal and external documents might be separated, or documents about the UNIX operating system and DCE. No additional levels are needed.

See Section 5.1.3 for the recommendations for the content model of bibliography lists.

3.2.7 Glossary

The tree diagram in Figure 3-5 shows the glossary.

The glossary division is a collector for definitions of terms used within the document. It contains the following, in order:

- A **title**
- Either zero or more elements from an IU1 mixture (that excludes glossary lists) followed by one or more **glossary subdivisions**, or a **glossary list**

A glossary subdivision contains the following, in order:

- A **title**
- A **glossary list**

Glossaries are likely to be manually created in some environments and created through augmentation in others. The presumption of augmentation has an effect on the recommended design for glossaries; the content model is more highly controlled than for bibliographies. (It has also had an effect on the recommended design for glossary lists.)

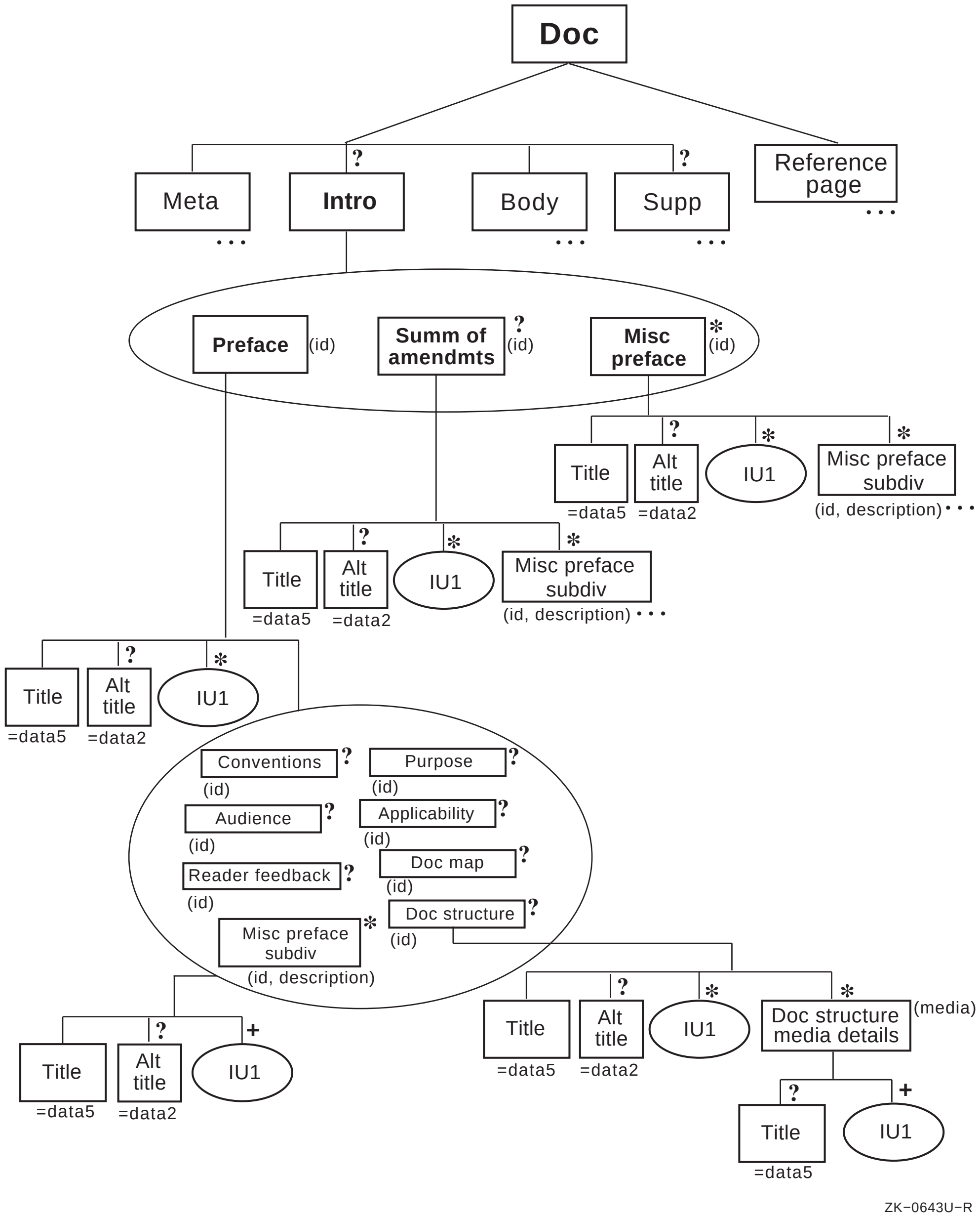
A glossary has one optional level of subdivision; for example, subdivisions could contain the terms starting with each letter of the alphabet. No additional levels are needed.

See Section 5.1.3 for the recommendations for the content model of glossary lists.

3.3 Introductory Information

The tree diagram in Figure 3-6 shows the introductory information.

Figure 3-6: Introductory Information



The introductory information, if present, contains the following in any order:

- A required **preface**, which introduces the document, its conventions, and its subject matter
- An optional **summary of amendments**, which details any changes made to the document since the last revision
- Zero or more **miscellaneous prefaces**, freely interspersed among the other two, which provide any additional introductory information (such as a summary of technical changes made to the product)

The preface contains a **title**, followed by an optional **alternate title**, followed by zero or more elements from an IU1 mixture, followed by a combination of the following in any order:

- Optionally, a specialized **applicability subdivision** that describes the applicable standards that the product adheres to
- Optionally, a specialized **audience subdivision** that describes the intended audience for the document
- Optionally, a specialized **conventions subdivision** that describes the typographical and other conventions the reader needs to understand the document
- Optionally, a specialized **document structure subdivision** that describes how the document is organized
- Optionally, a specialized **reader feedback subdivision** that describes how the reader can send document comments to the organization
- Optionally, a specialized **document map subdivision** that describes the document in relation to the collection it belongs to, and how the reader can best use the document set
- Optionally, a specialized **document purpose subdivision** that describes the document's function
- Zero or more **miscellaneous preface subdivisions** that cover additional topics, each with a required **description** attribute describing its contents, freely interspersed among the specialized subdivisions

The order is not fixed because the IAWG determined that, even though there is no flow in the arrangement of divisions, receivers would generally be willing to use any source order provided as the output order.

These subdivisions are unique elements because only one of each element can be allowed to be present; this aids retrieval (in the case of conventions, for example) and automatic suppression of information irrelevant to the receiver. The miscellaneous subdivision is for extensibility if additional distinct preface topics need to be supplied.

Restricting the occurrence of specialized preface subdivisions and giving flexibility to order them meets the needs of this information. However, prototyping revealed that the special content model required for the preface may contribute to some greatly increased capacities and quantities for the DTD. If basic restrictions on using the SGML language make this arrangement impossible, a simpler content model may be needed here.

For the miscellaneous subdivision and all the specialized subdivisions except document structure, the content model is as follows:

- A **title**
- An optional **alternate title**
- One or more elements from an IU1 mixture

The document structure subdivision is special in that it may need to describe different structures depending on the delivery form; for example, an online hypertext presentation of the document may need a different explanation from a printed presentation of the same document. Therefore, this subdivision contains the following, in order:

- A **title**
- An optional **alternate title**
- Zero or more elements from an IU1 mixture
- Zero or more specialized **document structure media details** regions, each of which has a **media** attribute for recording the delivery form that the contained description applies to, and contains an optional **title** and one or more elements from an IU1 mixture

  The title is optional because this element can be treated as a division, or merely as a special collector that can be used to conditionally include information based on the presentation instance. This is the only case in the introductory information where divisions are allowed to go down three levels.

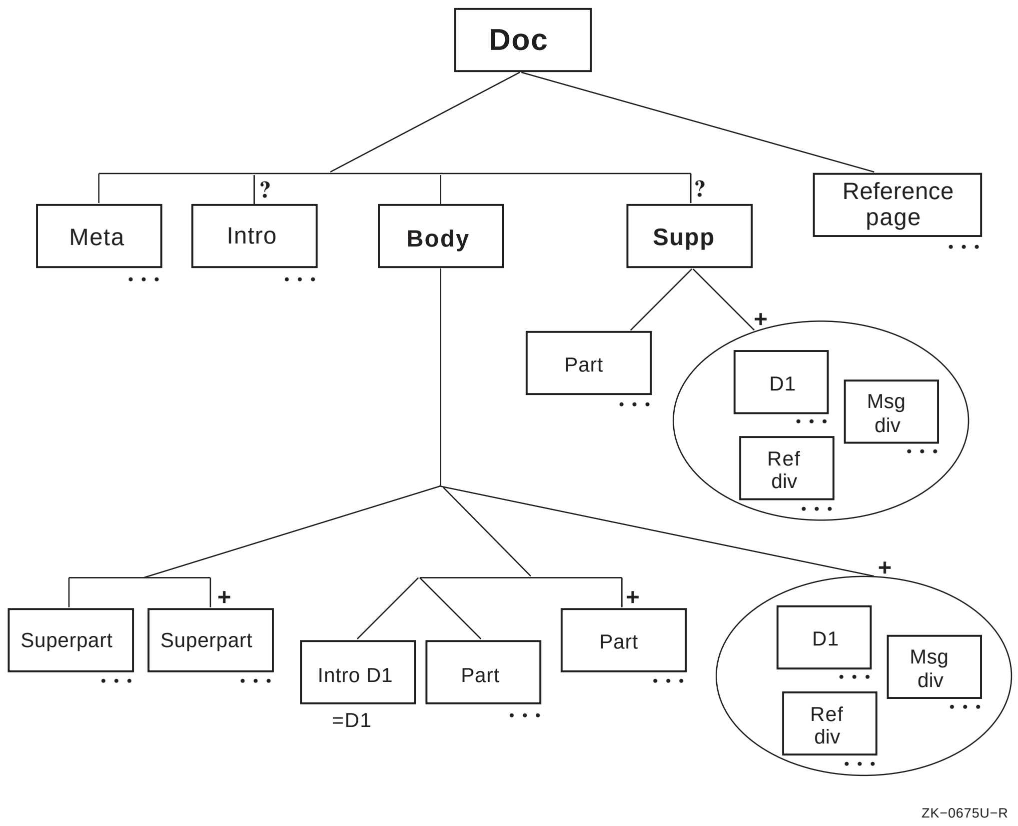
The summary of amendments and the miscellaneous preface both contain the following, in order:

- A **title**
- An optional **alternate title**
- Zero or more elements from an IU1 mixture
- Zero or more **miscellaneous preface subdivisions**, as described above

3.4 Document Body and Supplementary Information

The tree diagram in Figure 3-7 shows the body and supplementary information.

Figure 3-7: Document Body and Supplementary Information



In the following discussion, D*n* means a division at the specified level; for example, ''D3'' refers to a third-level division.

The document body can have one of three basic structures:

- Two or more **superparts**

  A minimum of two is needed to ensure that the use of superparts is not spurious.

- Either an **introductory D1** followed by one or more **parts**, or two or more **parts** without the introductory D1

  A minimum of two is needed unless an introductory division is present, which allows, for example, an introduction to a single part containing a reference division.

- One or more **D1s**, **reference divisions**, and **message divisions**, in any order

  In the context of the body, D1s function as chapters.

  A minimum of one is needed because it is possible to have a document body that contains a single major grouping of information; this is often the case with installation guides, or with reference manuals that contain a single specialized reference division. (The body is invisible, and cannot serve as the division component.)

If there is only one division, special formatting might be needed. For example, a singular chapter would not be numbered. If a formatter needs to know ahead of time whether there is only one, in-house preprocessing to add this information would probably be necessary.

The supplementary information has one of two basic structures:

- A single **part**

  This choice reflects the fact that this information is sometimes explicitly labeled as being in a supplementary grouping.

  The appearance of a part in supplementary information is its only use in a role functionally different from that of a regular part. Rather than creating a distinct element for it, or creating an alternative supplementary element that can contain a title and be ‘‘visible,’’ we felt that the best solution was to allow the part’s context to distinguish it from the regular parts.

- One or more **D1s** which, in the context of supplementary information, function as appendixes

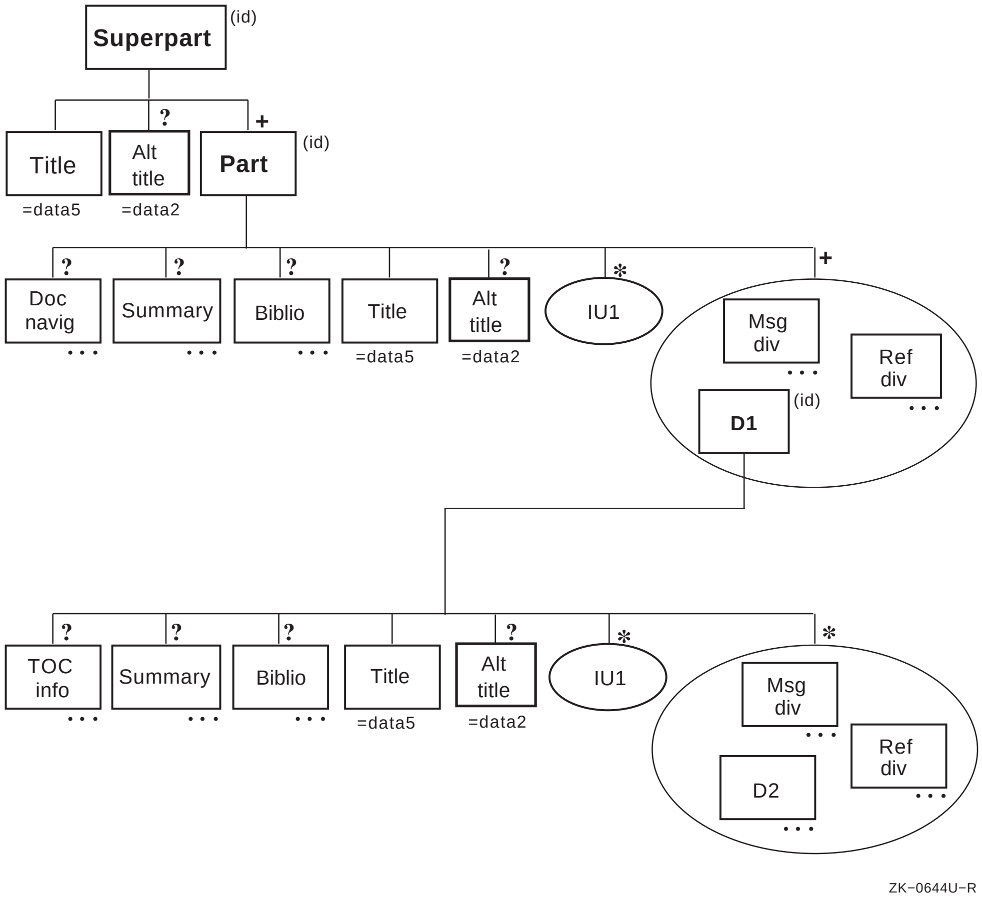
Superparts contain parts, parts contain D1s, D1s can contain D2s, and so on through D7. All these divisions have **identification** attributes, and the **localization status** and **localization comment** attributes, which are discussed in Section 2.2.10.

See Table 2-3 for a summary of the contents of the divisions within these containers.

3.4.1 Superparts, Parts, and D1s

The tree diagram in Figure 3-8 shows superparts, parts, and D1-level divisions.

Figure 3-8: Superparts, Parts, and D1s



A superpart contains, in order, a **title**, an optional **alternate title**, and one or more **parts**.

A part contains the following, in order:

- Optional **document navigation information**
- An optional **summary**
- An optional **bibliography**
- A **title**
- An optional **alternate title**
- Zero or more elements from an IU1 mixture
- One or more generalized **D1s**, **reference divisions**, and **message divisions**, in any order

D1 and the introductory D1 have the same content model; they contain the following, in order:

- Optional **navigation information**

- An optional **summary**
- An optional **bibliography**
- A **title**
- An optional **alternate title**
- Zero or more elements from an IU1 mixture
- Zero or more generalized **D2s**, **reference divisions**, and **message divisions**, in any order

Parts and D1s have basically the same content model, but according to the document model, a part is not expected to directly contain real document subject matter as D1 can; therefore, at least one D1, reference division, or message division is required inside it. For the same reason, specialized divisions are not allowed as peers to parts; rather, a part should contain a single specialized division at the D1 level.

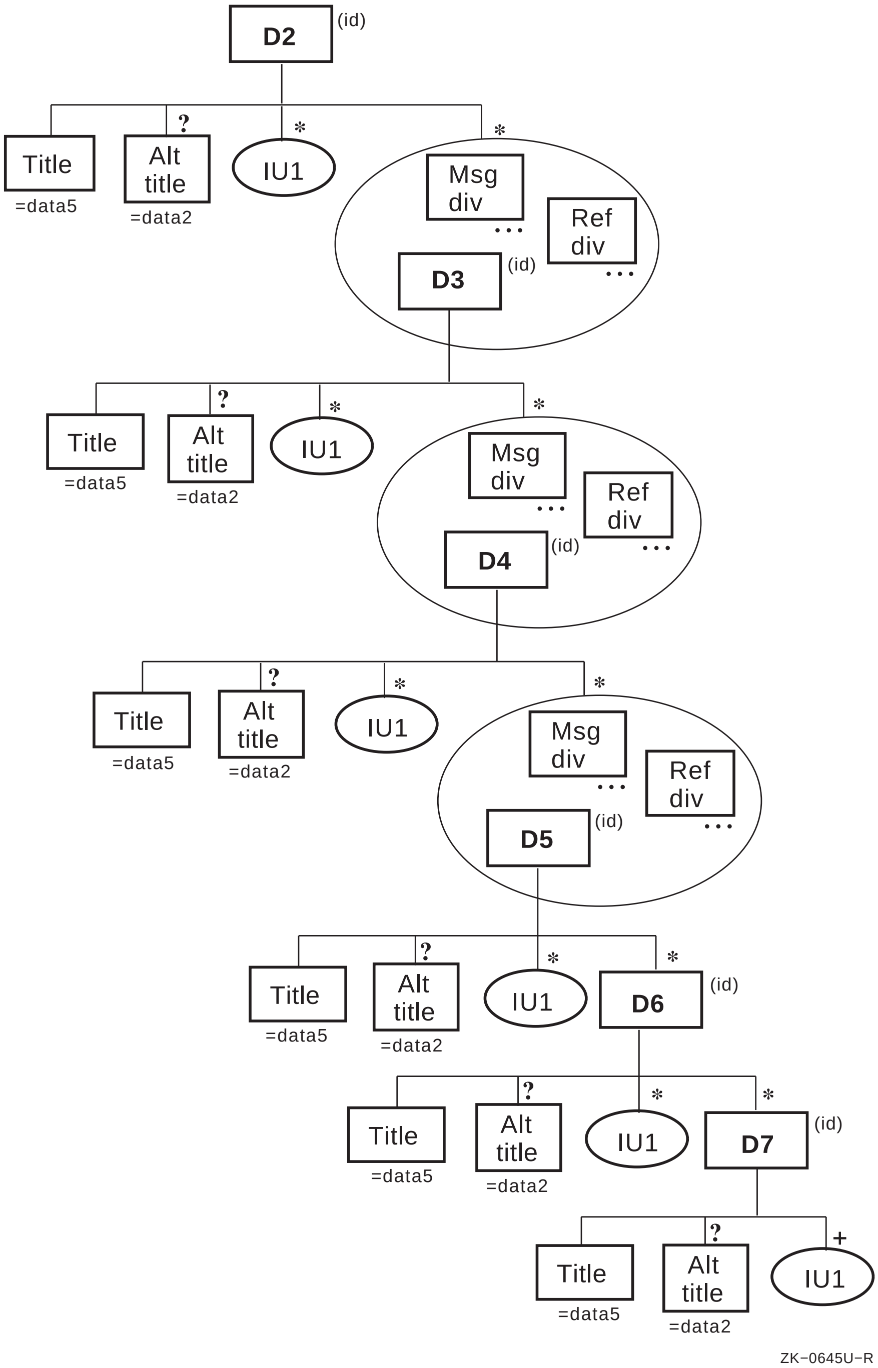
The document model dictates that D1 is the first level that is expected to contain subject-matter information directly (which is why we have called it a first-level division rather than a third-level division); as a result, it does not require any divisions inside it. Although it is usually considered poor organization, D1 can have a single subdivision in it; for example, in grid-pattern documents (where a set of subdivisions is provided in a template for repeated use, and each subdivision is used only where it applies), it is possible to legitimately use only one subdivision.

The introductory D1 has a different name from the regular D1 because, when the division is followed by parts, it has a functional role distinct from that of a regular chapter. If they had the same name, without knowing what follows the first chapter, an application would not be able to tell whether it is a regular division or an introductory one. In addition, having a separate element allows the possibility of having attributes and a content model unique to the introductory D1.

3.4.2 D2 Through D7

The tree diagram in Figure 3-9 shows the D2-level through D7-level divisions.

Figure 3-9: D2 Through D7



D2, D3, and D4 have the same basic content model; they contain the following, in order:

- A **title**
- An optional **alternate title**

- Zero or more elements from an IU1 mixture
- Zero or more **subdivisions** the next level down, **reference divisions**, and **message divisions**, in any order

D5 and D6 have the same basic content model; they contain the following, in order:

- A **title**
- An optional **alternate title**
- Zero or more elements from an IU1 mixture
- Zero or more generalized **subdivisions** the next level down

D7 has the simplest content model of all the generalized divisions:

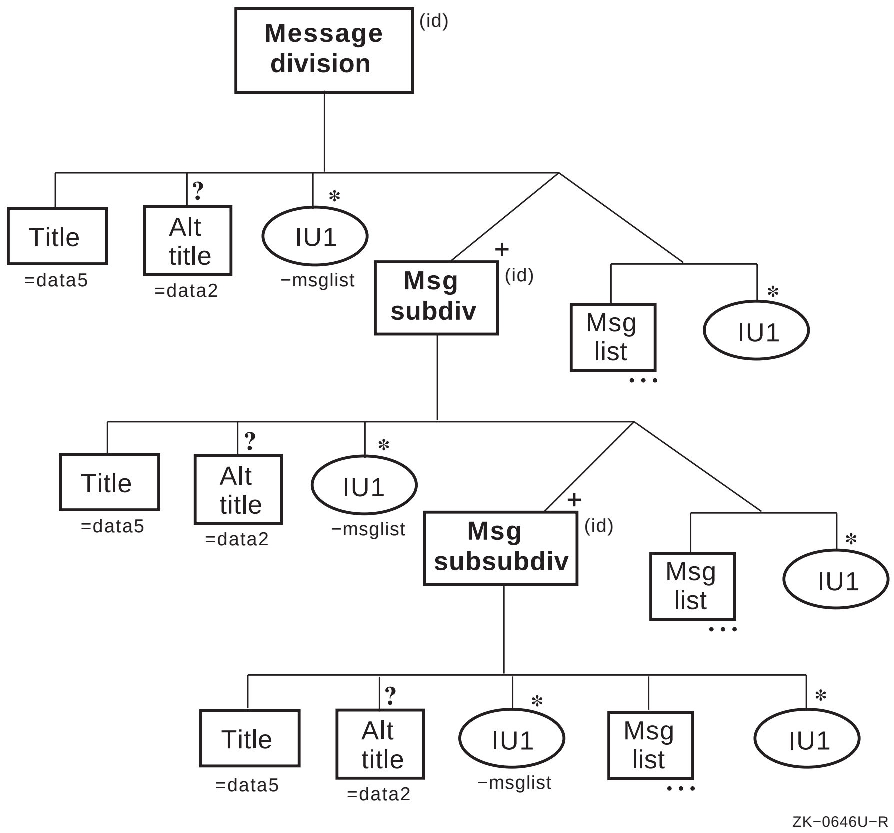
- A **title**
- One or more elements from an IU1 mixture

D5, D6, and D7 do not contain the reference and message divisions; these specialized divisions allow two levels of specialized subdivision within them, and if they are used inside the D4 level, they already offer the ability to go down two more levels.

3.4.3 Message Divisions

The tree diagram in Figure 3-10 shows the message divisions.

Figure 3-10: Message Divisions



Message divisions allow the grouping of messages by function and the ability to access and manipulate those groupings and their introductory material. Message lists are allowed in generalized divisions by themselves, but if generalized divisions are used to group message lists, the function groupings and their introductory text cannot be found and used in any special way.

A message division contains the following, in order:

- A **title**
- An optional **alternate title**
- Zero or more elements from an IU1 mixture that excludes message lists (see Section 2.2.1 and Section 2.2.3 for information on some consequences of this)
- Either one or more **message subdivisions**, or a **message list** followed by zero or more elements from an IU1 mixture (which can include additional message lists)

A message subdivision contains the following, in order:

- A **title**
- An optional **alternate title**
- Zero or more elements from an IU1 mixture that excludes message lists
- Either one or more **message subsubdivisions**, or a **message list** followed by zero or more elements from an IU1 mixture (which can include additional message

lists)

A message subsubdivision contains the following, in order:

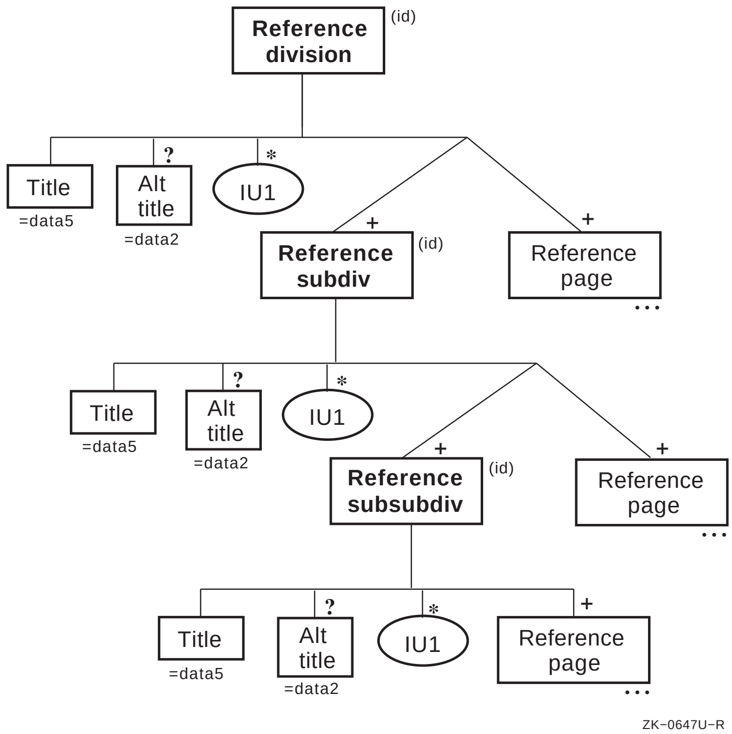
- A **title**
- An optional **alternate title**
- Zero or more elements from an IU1 mixture that excludes message lists
- A **message list** followed by zero or more elements from an IU1 mixture (which can include additional message lists)

A message division has two optional levels of subdivision because, for example, messages for different software modules might be separated, and within them, messages that have different severity levels. No additional levels are needed.

3.4.4 Reference Divisions

The tree diagram in Figure 3-11 shows the reference divisions.

Figure 3-11: Reference Divisions



Reference divisions are the containers for multiple reference pages in a manual; they allow the grouping of pages by function and the ability to access and manipulate those groupings and their introductory material. No other context, besides the top level of the document type, contains reference pages.

A reference division contains the following, in order:

- A **title**

- An optional **alternate title**
- Zero or more elements from an IU1 mixture
- Either one or more **reference subdivisions** or one or more **reference pages**

A reference subdivision contains the following, in order:

- A **title**
- An optional **alternate title**
- Zero or more elements from an IU1 mixture
- Either one or more **reference subsubdivisions** or one or more **reference pages**

A reference subsubdivision contains the following, in order:

- A **title**
- An optional **alternate title**
- Zero or more elements from an IU1 mixture
- One or more **reference pages**

A reference division has two optional levels of subdivision because, for example, reference pages for different software modules might be separated, and within them, reference pages for different audiences. No additional levels are needed.

Reference Page Organization Recommendations 4

This chapter describes the recommendations for the organization of reference pages. See Chapter 3 for recommendations for the allowable contexts of reference pages, Section 2.2.6.1 for general discussions of major reference page themes, and Section 1.3.12 for recommendations on the relation of this document type to the Committee for Common Man (CFCM) work.

The tree diagrams in this chapter (whose notation is explained in Figure A-1) provide a useful overview of the recommendations, but the recommendations are explained completely only in the text. Where no indication of element occurrence is given in the text (for example, ‘‘zero or more’’), the element is assumed to be required. Also, attributes are less completely specified than elements; their names, types, possible values, and defaults are typically implied by the accompanying explanation.

See Table 2-1 and Table 2-2 for an explanation of the IU and data mixtures found in various contexts in the reference page organization.

The recommended reference page document type encompasses both UNIX man pages and modules of software reference information that do not use the `man` tools for their presentation. The two are nearly identical in function, scope, and structure. The name ‘‘reference page’’ is intended to encompass both.

We have recommended a single document type for command, routine, file, and all other reference pages. They are not sufficiently different from each other to warrant having distinct document types; it would be difficult to enforce the usage of the correct one for each documented software component. However, collapsing the few differences into a single document type has some consequences. For example, in a file reference page, no usage model need be present. Therefore, the usage model has been made optional.

4.1 Metainformation

Reference pages appear to have fewer needs for metainformation than do manuals. Much information in reference pages that seems to be metainformation-like is actually descriptive of the software entity being documented; for example, history sections discuss technical changes to the software, not the revision history of the reference page. Also, the document model for reference pages expects some metainformation to be supplied for direct reader consumption, and this information is treated like a normal part of reference page subject matter (discussed in Section 4.7).

Attributes can probably accommodate all the necessary metainformation for reference pages; if they cannot, a simplified manual metainformation component might be added to reference pages. Alternatively, if the regular manual metainformation component is well suited to reference pages, the very top level of the document type organization can be modified to allow reference pages as well as manuals to include it.

A reference page has at least the following attributes, all related to metainformation:

- The **identification** attribute uniquely identifies the page for reference from elsewhere.
- The **category** attribute describes the type of information documented in the page, such as user command, administrator command, system call, library routine, file, device, statement, and tag. For pages used with `man`, this attribute will indicate the traditional section that the page belongs to (for example, 1m or 8 for administrator commands); however, the tokens in the list should be named abstractly, labeling the kind of information without indicating a section number.
- The **audience** attribute can take a list of descriptors, such as user, administrator, and programmer.
- The **validity date** attribute gives the date the reference page is in final form, ready for use.
- The **version** attribute gives the system and version to which the page applies.

4.2 Top-Level Organization

The tree diagram in Figure 4-1 summarizes the top-level reference page organization.

Just as a manual's subject matter is grouped into core and supplementary information, a reference page's subject matter has basic groupings.

The reference page has the following high-level information containers, in order. (Some of these containers were originally formulated by CFCM as high-level divisions; the design group considers them invisible collectors for related information, to be used only for localized content model control and information retrieval.)

- Synoptic or **brief** information on the software being documented
- A reference page **body**, which describes in detail the intended function of the software being documented and supplies any usage notes, examples, and information on interactions with other systems

  This container was originally divided by CFCM into specification information and usage information (similar to the distinction between normative and informative information in standards documents). However, we have discovered that in practice, the two kinds of information are completely intermixed and it is difficult to identify which category some components belong to. For example, descriptions of subcommands of an interactive command both describe the usage of the main command and give a specification of its features.
- Optionally, **reference** information, which points to other relevant sources of information
- Optionally, contextual or **''about''** information, which contains miscellaneous descriptive or status information about the software entity

Some of the high-level containers are optional because the information in them is not required to be present. When an optional high-level collector is present, it cannot be empty.

The elements inside the body, references, and about containers each cover a single well-defined subject area. In the typical reference page presentation, the subject areas are formatted to look like divisions. However, they are less like manual divisions than like fields in a reference-information database record because their content is highly specialized by the reference page document model. Therefore, they are referred to here as subject areas instead of divisions.

4.3 Subject Areas

The document model for the subject areas in the body, references, and about containers indicates that they are flat and modular. No subject area contains another; they do not depend on being in a flow that adds intelligence (their output ordering being determined by the page's category attribute or by an organization's stylistic preferences); and each one is considered to contain the complete set of information for its topic, using cross-references as necessary to point to other relevant information.

Given these conditions, we applied the component ordering principle to recommend a fixed source order of OSF's choosing to be used for the subject areas. This gives syntactic predictability to received documents, allowing receivers to reorder the subject areas without needing to develop applications that accommodate all the combinations of input and output orders.

In addition to ordering subject areas according to house style rules, organizations also label subject areas differently. For example, the same subject area is labeled ''Parameters'' by some and ''Arguments'' by others. Because such a subject area has a unique element, organizations can apply their preferred title to the area during formatting.

Some subject areas may need additional structure inside them. This can be accomplished with either generalized subdivisions or bridging titles, depending on whether hierarchical structure is seen as necessary. However, the fact that each subject area is modular suggests that bridging titles are more appropriate, because they keep the module intact. In the traditional linear presentation style, subject areas would correspond to `.SH`-level sections and bridging titles or subdivisions would correspond to `.SS`-level sections.

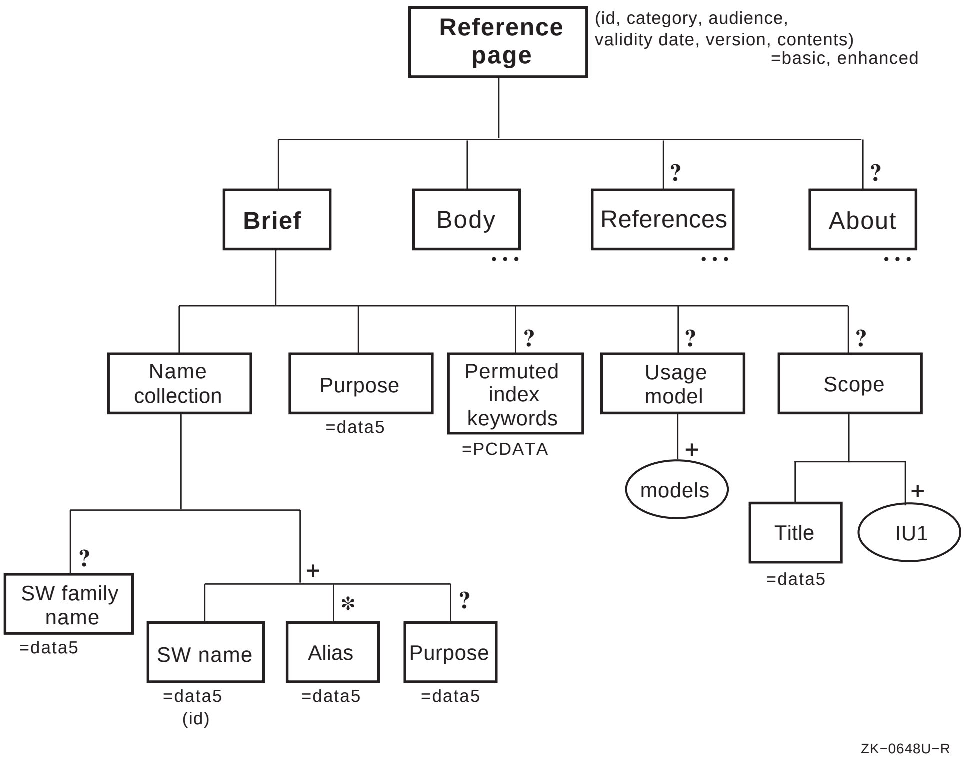
Within each high-level container, **miscellaneous subject areas** can appear after the specialized subject areas, to cover subject areas for which a need was not foreseen. The specialized subject areas contain one or more elements from an IU1 mixture, and the miscellaneous subject area contains a required **title** and one or more elements from an IU1 mixture, as well as having a required **description** attribute. (These content model mixtures assume the bridging title solution is being used.) Any new OSF offering would probably need a customized list of description values, which would have to be institutionalized in a coding style guide until a revised DTD version could create elements for the new subject areas.

Most of the subject areas recommended here were originally specified by CFCM as typical reference page topics; CFCM should be able to supply more precise descriptions for these components than appear here.

4.4 Brief Information

The tree diagram in Figure 4-1 shows the brief information.

Figure 4-1: Reference Page Organization



The contents of the brief information are highly structured; this high-level container functions like a prolog to the rest of the page.

- The **name collection** identifies the software entity or entities being described by the page.
- The **purpose** is a concise description of the function of the entity or entities, similar to an abstract. It contains a data5 mixture.

  Note that the design group considered adding an additional ‘‘extended’’ purpose, similar to a summary, but finally concluded that the other available descriptors are sufficient because the document model for reference pages assumes a single page is fairly small, typically smaller than a chapter in a manual (which allows summaries).
- The optional **permuted index keywords** element supplies the descriptor information to be used in the construction of permuted indexes. It is somewhat similar to the retrieval keywords metainformation for manuals. It contains PCDATA.
- The optional **usage model** contains one or more elements from the model class; models describe how to use the software entity or entities. (See Section 5.1.8 for more information on models.) This container element is needed so that all the usage models inside it can be operated on as a group. Most reference pages are expected to contain at least one model, but pages that describe files do not need to provide such information. No IUs other than models are allowed here because

extended text descriptions belong in the body.

The information on usage models is usually formatted as a division with the title ‘‘Syntax’’ or ‘‘Synopsis’’. The title chosen depends on the stylistic preference.

Individual syntax models can have an optional **identification reference** attribute that indicates the primary name to which it applies, if there are several.

- The optional **scope** element contains brief system availability, location, or object-oriented class information, as appropriate to the software entity. It contains a **title** that identifies its contents, and one or more elements from an IU1 mixture.

  This information can be moved on output to the ‘‘about’’ part if desired.

The name collection contains an optional **family name**, which is a name, other than one of the entities’ real names, to be applied to all the software entities documented in the page. The family name might be printed on the page as a subcategory name, or it might be used in retrieval. It contains a data5 mixture.

The information following the family name is a required repeatable block. One or more primary software entity **names** must be supplied, each along with any **alias** names that accomplish the same result, and an optional **purpose** description specific to that name that supplements the main purpose description. These individual purpose descriptions can be interspersed with the usage models on output. All the elements in the block contain a data5 mixture.

An example of three primary names discussed in one page is `grep`, `egrep`, and `fgrep`; they are closely related, but accomplish different tasks. An example of a primary name and an alias is `apropos` and `man -k`. Aliases are not usually listed along with primary names in the traditional presentation of reference pages; they could be used as a search function during online access rather than necessarily being given the same prominence as the primary name at the top of a formatted page.

Each primary name can have an **identification** attribute so that throughout the page, discussions of features that apply specifically to one name or another can be linked to the applicable name. These links are described along with the components they are attached to.

4.5 Body Information

We recommend at least the following elements for specialized subject areas in a fixed order. (Note that the following order is just a suggestion and any other fixed ordering would be acceptable.)

- **Description** subject area for general description of the software’s features
- **Parameters** (or arguments) subject area
- **Options** (or flags or switches) subject area
- **Keys** subject area
- **Fields** subject area
- **Subcommands** subject area for descriptions of commands available within an interactive command’s environment
- **Types** (or data types) subject area

- **Functions** subject area for the description of the software's functions
- **Results** subject area
- **Return codes** (or return values) subject area
- **Macros** subject area
- **Constants** subject area
- **Files** subject area for the description of any files related to the software (including the main file name for reference pages that are about files)
- **Restrictions** subject area for the description of any restrictions on using the software, particularly in combination with other software or systems
- **Limitations** subject area for the description of any expected capabilities that the software does not have
- **Examples** subject area

4.6 References Information

References information currently contains a specialized **see also** (or related information) subject. This high-level container could be used in the future to contain additional subject areas that perform hypertext functions, such as containing a ''web'' of related pages that form a single hyperdocument.

Cross-references to outside documents are not disallowed from this division, or from reference pages in general. If they exist in an instance and contain references to documents or reference pages that receivers do not have available, the references can at least be found and dealt with because cross-references to outside documents and other reference pages are different components from an internal cross-reference.

4.7 ''About'' Information

We recommend at least the following elements for specialized subject areas in the ''about'' high-level container; note that the following order is just a suggestion and any other fixed ordering would be acceptable:

- **Software acknowledgments** subject area for acknowledgments to people or organizations that participated in the development of the software
- **Software security** subject area for the description of security features and standards related to the software
- **Software license** subject area for the description of licensing requirements
- **Software authors** subject area for the description of the individual developers of the software
- **Software history** subject area for the description of the history of the software's development

- **Copyrights** subject area for any copyright notices related to the software or the reference page
- **Trademarks** subject area for the any trademark attributions related to the software or the reference page

Following these subject areas, zero or more **miscellaneous subject areas** should be allowed.

Information Pool Recommendations 5

This chapter describes the recommendations for the components in the information pool.

The tree diagrams in this chapter (whose notation is explained in Figure A-1) provide a useful overview of the recommendations, but the recommendations are explained completely only in the text. Where no indication of element occurrence is given in the text (for example, ‘‘zero or more’’), the element is assumed to be required. Also, attributes are less completely specified than elements; their names, types, possible values, and defaults are typically implied by the accompanying explanation.

See Table 2-1 and Table 2-2 for an explanation of the IU and data mixtures found in various contexts in the information pool.

5.1 Information Units

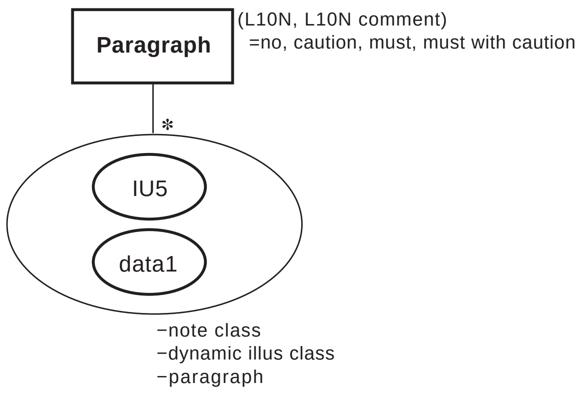
Information units (IUs) make up the main content of divisions and some metainformation containers, and some of them can appear directly or indirectly inside other IUs as well.

The following sections give the recommendations for the IUs.

5.1.1 Paragraph

The tree diagram in Figure 5-1 shows the paragraph.

Figure 5-1: Paragraph



ZK−0663U−R

A **paragraph** is a primarily text-based discussion of a topic that is limited in scope. It makes up most of the information content of manuals.

Section 2.2.1 discusses our recommendations for the content of paragraphs. A paragraph contains a mixture of the data1 and IU5 mixtures, excluding paragraphs themselves, the note class, and the dynamic illustration class. (Note that when a paragraph is used inside an IU2, IU3, or IU4 mixture, its content is restricted to a mixture of data1 and the generalized list class.)

A paragraph has the attributes **localization status** and **localization comments**, which are described in Section 2.2.10.

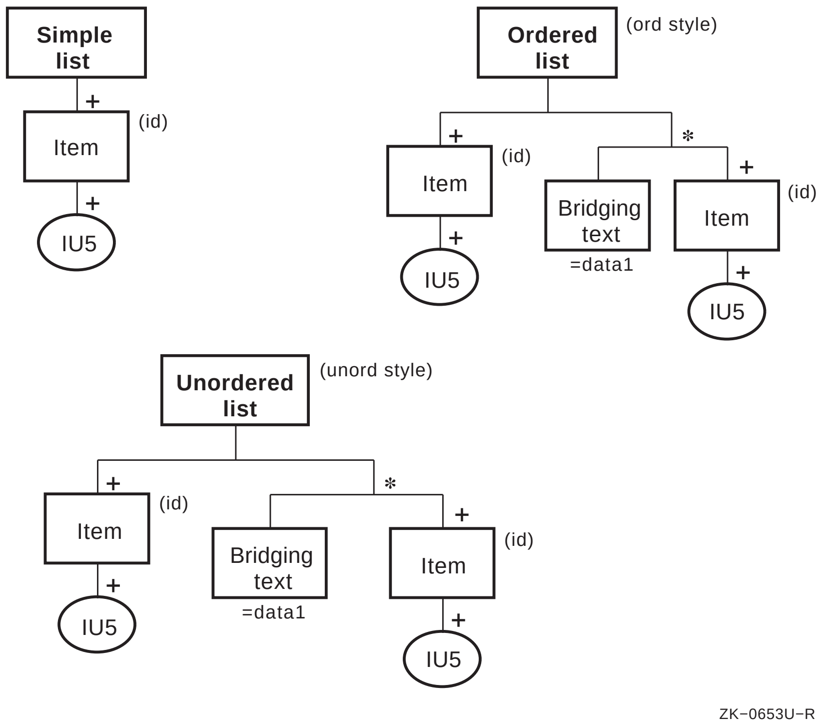
5.1.2 Generalized Lists

The generalized lists are enumerations of material that can contain any information of the writer's choosing.

5.1.2.1 Simple List, Ordered List, and Unordered List

The tree diagram in Figure 5-2 shows the simple, ordered, and unordered lists.

Figure 5-2: Simple, Ordered, and Unordered Lists



A **simple list** is a simple enumeration of items of information, typically without any special presentation such as item ‘‘bullet’’ indicators. It contains one or more **items**.

An **unordered list** is an enumeration of items whose presented order is insignificant, and an **ordered list** is an enumeration of items whose presented order is significant. The type of order can be by chronology, priority, some other criterion, or even for convenience of reference from the text; no distinctions are made between order types. These two lists have the same contents: one or more **items**, optionally interspersed with **list bridging text**, which is a specialized paragraph-like element containing a data1 mixture. It exists to allow, for example, descriptions of conditions that apply to items that follow; allowing bridging text, rather than forcing the list to be ended and started again after a paragraph, ensures that the list can be manipulated as a whole.

Bridging text is disallowed at the beginning or end of a list, and only a single block of bridging text can occur between two items. However, procedure bridging text – which is similar in function to list bridging text – allows complete interspersing of the text with the procedure steps. The content model of lists may benefit by such a broadening of the contexts in which list bridging text is allowed, for consistency.

The items in all three lists are identical, for markup consistency. An item contains one or more elements from an IU1 mixture and can have an **identification** attribute for reference from elsewhere (for example, a cross-reference from the text). An individual item cannot override the style or numbering dictated by its containing list.

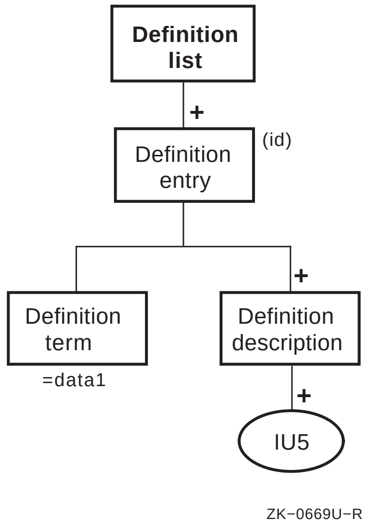
The unordered and ordered lists are different elements to allow separate sets of attributes. An unordered list has an **unordered style** attribute that has an ENTITY type, for the specification of a character or symbol to use as a list item ‘‘bullet’’ indicator. (An ENTITY type allows the set of styles to expand.) An ordered list has an **ordered style** attribute that has a list of token values: at least roman uppercase, roman lowercase, arabic, alphabetic uppercase, and alphabetic lowercase. We recommend a token list because the possible set of styles is limited. The attribute values should be implied; under normal circumstances, we are assuming that formatting applications can choose the correct style at different nested levels of list. These attributes are presentational in nature, but need to be supplied because sometimes the choice of style carries meaning; writers need to be able to control it if necessary.

All three lists allow a minimum of one item, so that lists used in a grid pattern (where a set of list items is dictated in a template for repeated use, and each item is used only where it applies) can properly include only a single item.

5.1.2.2 Generalized Definition List

The tree diagram in Figure 5-3 shows the generalized definition list.

Figure 5-3: Generalized Definition List



A **generalized definition list** is a listing of terms, each with an informal definition of their meaning or description of their significance. This component is recommended in addition to segmented item lists; while a two-segment list might possibly have the same presentation, it would not be able to indicate that it contains definitions in particular. This component is also different from a glossary list, which contains official definitions of vocabulary.

5.1.2.3 Branching List

A possible generalized list component for branching lists was uncovered at the OSF Documentation SIG's June 1992 meeting. Branching lists are similar to generalized definition lists. They consist of items that contain a main term and a description. The description consists of two or more ''tracks,'' each of which describes a possible value for the main term; further descriptions might have additional tracks.

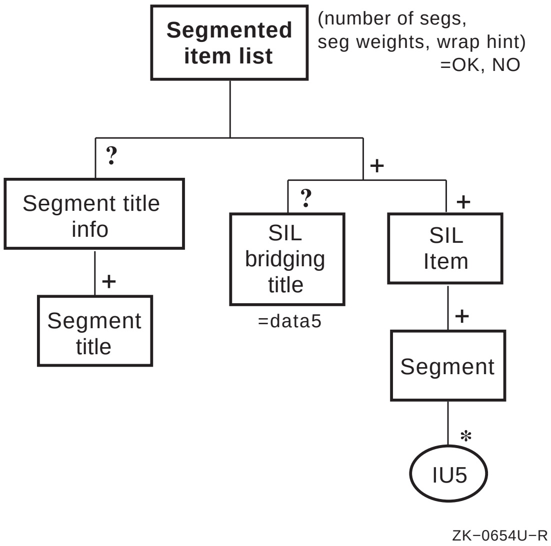
For example, an item of a branching list describing binary variables might offer an explanation of each true and false value for that item's variable in different tracks.

The design group has not made any specific recommendations for branching lists, but OSF may determine that having this component distinct from other generalized lists is useful.

5.1.2.4 Segmented Item List

The tree diagram in Figure 5-4 shows the segmented item list (SIL).

Figure 5-4: Segmented Item List



A **SIL** is an arrangement of information that is logically two-dimensional. It is like other lists in that it contains a series of items, but the items each have a number of segments; they are *n*-tuples.

There are many SIL-type components in these recommendations, such as the specialized and generalized definition lists. However, these all have some specialized meaning attached to the segment positions; definition list entries have a ‘‘term’’ position and a ‘‘description’’ position, for example. A SIL is generalized, so that the segment positions have meanings determined solely by the writer.

It can be useful to think of a SIL as a simple case of a table. An ASCII character lookup table is an example of a SIL: each item (row) in the list has two segment (column) positions, one for a character code and one for the symbol represented by the code. Each of the segment positions has a title (column heading).

A SIL is allowed on its own as an IU, and also as one of the encodings for information in a table; Section 5.1.9 discusses the meaning of the table context in more detail. When it is not in a table, the assumption is that formatting applications will probably not format it tabularly, but rather as a list with structured items (as message lists are typically handled). Segment titles could be repeated within each instance of an item down the list.

For example, the following might be how a SIL looks when inside a table:

Variable	Type
a	Boolean
b	String

The same SIL when outside a table might look as follows:

Variable: a

Type: Boolean

Description: Turns the ABC characteristic on or off.

Variable: b

Type: String

Description: Sets the value of the XYZ characteristic.

Note that no logical limit can be set on the number of segment positions; if the SIL is thought of as a forms database, the number of fields in the form can be arbitrarily large. (Tabular formatting typically imposes some presentation limits, however.)

Many of the tables interchanged with OSF can be encoded as SILs. Using a SIL component and producing tabular output from it can solve many of the table interchange problems faced by OSF interchange partners because it simplifies many processing assumptions. Wherever some information cannot be handled by a SIL, however – for example, when three or more dimensions of information force a table to have vertical or horizontal spans – a more complex table encoding should be available.

A SIL contains the following, in order:

- An optional **segment title information** element, which contains one or more **segment titles**
- One or more **SIL items** interspersed with any number of **SIL bridging titles**

  The SIL bridging title exists to allow informal organization of the items, just as regular bridging titles do in text. It contains a data5 mixture.

A SIL item contains one or more **segments**, which each contain zero or more elements from an IU5 mixture.

The segmented item list has several attributes:

- A required **segment count** attribute that records the number of segments supplied in the items of this list instance

  This attribute can facilitate processing, but cannot control parser validation of the number of segments supplied in each item.

- A required **segment weights** attribute for supplying the relative weight or importance each segment cross-section could be given

  These values can be used as a series of proportion hints for contexts where the SIL is formatted tabularly. This is clearly presentational information, but needs to be controlled by writers. We have tried to cast it more abstractly, so that application developers will not feel unnecessarily constrained to interpreting this as fixed formatting information.

- A **wrap hint** attribute for giving applications permission to wrap tabularly formatted lists to a multiple of the supplied number of segment positions (in other words, formatting two or more entire items next to each other horizontally)

  This is another presentational attribute. It has a token value: either OK (the default) or NO. Wrapping is often desirable when the SIL contains many brief items, making lookup difficult. However, some items can easily be misinterpreted when presented this way, The attribute exists to let the writer indicate that wrapping the list is unacceptable. It does not, however, let the writer dictate any wrapping characteristics; this is assumed to be entirely in the application's domain.

5.1.3 Specialized Definition Lists

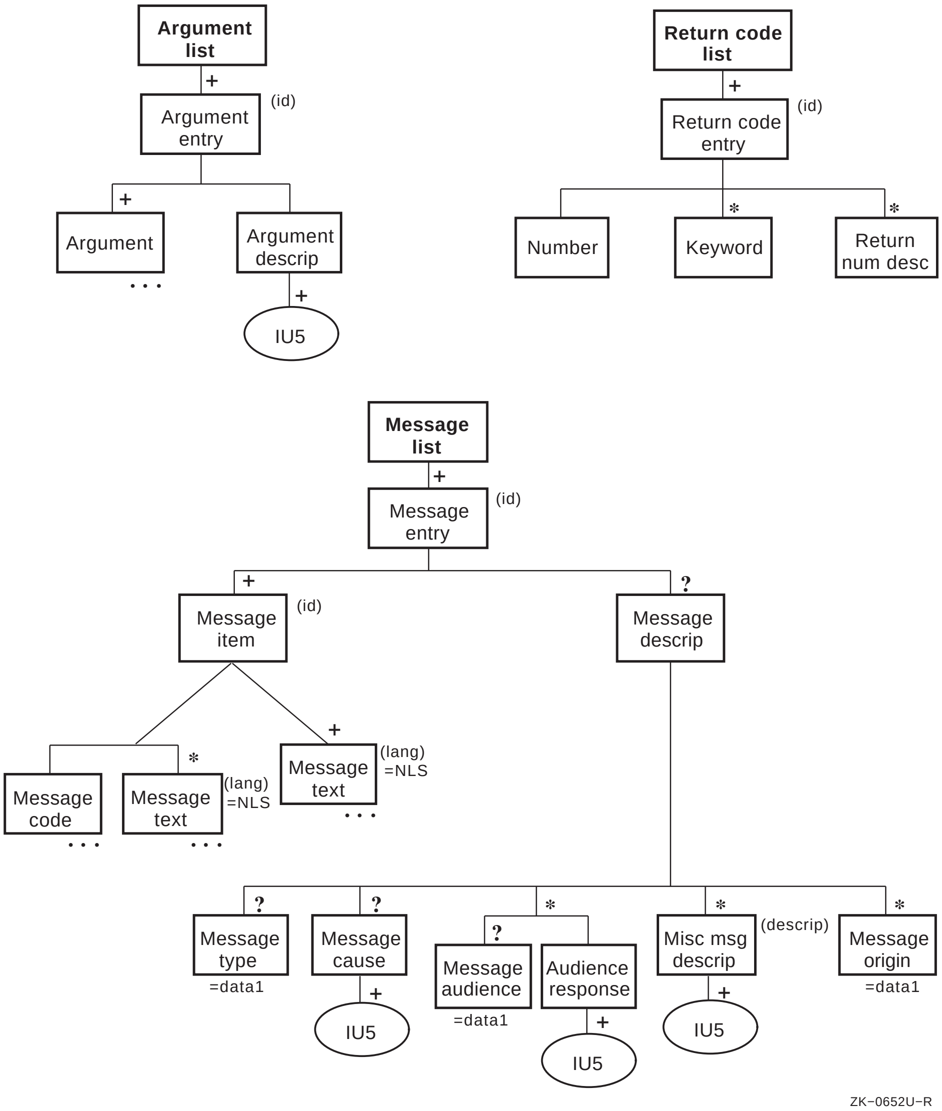
The following sections describe the recommendations for specialized definition lists.

Note that for some kinds of definitions, a single description may apply to more than one item (argument, message, and so on). In these cases, a boilerplate description can be written once, stored as a general entity, and referenced in all the places where it applies.

5.1.3.1 Argument List

The tree diagram in Figure 5-5 shows the argument list.

Figure 5-5: Argument, Return Code, and Message Lists



An **argument list** describes a collection of computer language arguments or parameters, such as arguments to commands or routines. It contains one or more **argument entries**, each of which can have an **identification** attribute for reference from elsewhere (for example, an argument that is mentioned in the text or that appears in a model).

An entry consists of one or more **arguments** (the same as the technical key data element), followed by an **argument description**, which contains one or more elements from an IU5 mixture (which can contain further argument lists). Indirectly nested argument lists are allowed because an argument can contain, for example, a command option and further arguments; the list that describes it must be able to contain another argument list.

See Section 5.1.8 for information about how argument lists relate to models.

5.1.3.2 Return Code List

The tree diagram in Figure 5-5 shows a possible organization for a return code list. However, further study and testing are needed in this area.

A **return code list** describes a collection of return values and codes.

As a starting point for discussion, the design group recommends that a return code list contain one or more **return code entries**, each of which has an **identification** attribute for reference from elsewhere and contains the following components:

- A unique return **number**
- Zero or more mnemonic return **keywords**
- Zero or more **return number descriptions**

5.1.3.3 Message List

The tree diagram in Figure 5-5 shows the message list.

A **message list** describes a collection of computer messages. It contains one or more **message entries**, each of which can have an **identification** attribute for reference from elsewhere (for example, links from a message code or message text string mentioned in the text). A single message entry contains one or more **message items** followed optionally by a **message description**. An entry can contain more than one item because some items are so closely related that they have identical descriptions.

Each message item can have its own **identification** attribute for reference from elsewhere; it is at the writer's discretion to point to a particular message item in an entry versus a whole entry that might contain multiple message items. (We have not recommended an additional container, beyond the message entry, for series of message items because there are already many container elements in message lists. Therefore, an extra processing burden is assumed because lookahead is required to determine if there are multiple items that have to be manipulated – for example, numbered.)

A message item is a container for all the information that identifies a single unique message. It contains a **message code** (a number or coded string unique to the message) if there is one, followed by versions of a **message text** string, in any languages in which it is supplied in the interface software, and possibly in any other languages into which the information creators have translated it. If the message has no code, at least one text string is required; if the message has a code, no text strings are required to be present. The message code and text are the same as the technical key data elements.

The message text has a **language** attribute indicating what language the text is in; the value should be a token list of national language support (NLS) codes. (See the Glossary for more information.) The message text string with the language attribute corresponding to that of the product ID element in metainformation is the primary string intended to be presented to the reader; the others can also be output if desired. This attribute mechanism avoids problems when several different languages are supplied within the interface software, but places a bigger burden on applications than if a single ‘‘primary’’ message text component were identified within the message list itself.

A message description contains several distinct pieces of information, which we recommend be in a fixed order to meet the ordering principle. Individually the pieces of information are optional, but if the description is present it cannot be empty. It contains the following optional elements, in order:

- A **message type** that describes the message's type or severity level; it contains a data1 mixture.
- A **message cause** that describes the reasons why the message could have been output; it contains one or more elements from an IU5 mixture.
- A repeatable pair consisting of a **message audience** containing a data1 mixture and an **audience response** containing one or more elements from an IU5 mixture. Each pair is directed at a different expected audience of the message. For example, a particular message might require a user to notify the system administrator, and the system administrator to change a system setting. There would be two pairs of elements; the first pair specifies ''User'' as the audience and describes the user response, and the second specifies ''System administrator'' as the audience and describes the administrator's response.

  We recommended two elements, rather than creating a distinct element for each type of audience, because the set of possible audience types is not closed. If a standardized list of needed audience types can be developed, however, an attribute on a single audience response element, with a list of token values, might work also. This would ensure that audience description text can be consistently generated.

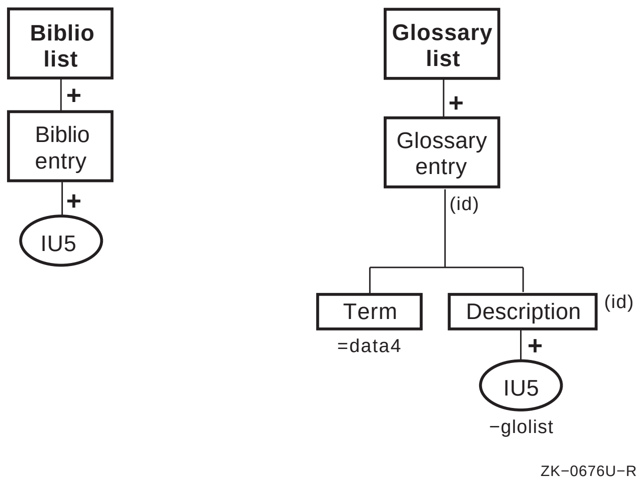
  We avoided creating an additional container element to hold each pair. The typical formatting case, where the audience description is treated as a title on the response description, imposes no special burden, but other processing, such as retrieval based on an audience search parameter, might be more complicated.
- Any number of **miscellaneous message descriptions**, containing one or more elements from an IU5 mixture, to accommodate distinct pieces of message description that do not already have unique components. It has a required **description** attribute for identifying the contents.
- Any number of pieces of **message origin** information, containing a data1 mixture, for identifying the software modules that produce this message.

There is no need to provide for multiple versions of the description in different languages; because the description is purely part of a document, rather than being part of the software interface, it is assumed to be just like the other text in the document.

5.1.3.4 Bibliography List

The tree diagram in Figure 5-6 shows the bibliography list.

Figure 5-6: Bibliography and Glossary Lists



A **bibliography list** contains one or more **bibliography entries**, each of which provides source information for a single document.

An entry contains one or more elements from an IU5 mixture that excludes bibliography lists. This exclusion is because the entries in a bibliography list should be modular. Where specific document titles and authors are mentioned in the data in these elements, the general key-data elements for document citation and document author are expected to be used.

An entry has an **identification** attribute for reference from elsewhere (for example, links from document citations in the text).

We have recommended this highly generalized bibliography entry because in our scope, bibliographies are typically informal and may include descriptions of the documents they list.

Bibliography lists are allowed in text as well as being mandated in document and division bibliographies; presumably such lists would be manually created, not augmented.

5.1.3.5 Glossary List

The tree diagram in Figure 5-6 shows the glossary list.

A **glossary list** contains one or more **glossary entries**, each of which provides one or more definitions for a single term.

A glossary entry has an **identification** attribute for reference from elsewhere (for example, links from the terms when used in text). It contains a **term** and one or more **descriptions**, in that order. A term contains a data4 mixture, which allows a graphic data element for icons or trademarked terms that have a graphic in them. A description contains one or more elements from an IU5 mixture that excludes glossary lists. This exclusion is because the entries in a glossary list should be modular.

Each description can have its own **identification** attribute for reference from elsewhere; it is at the writer's discretion to point to a particular definition in an entry versus a whole glossary entry that might contain multiple definitions.

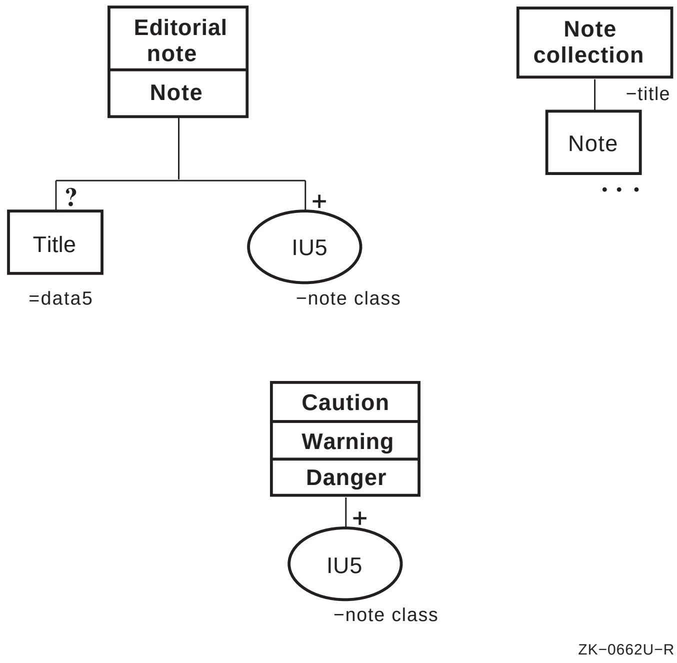
We have not recommended an additional container, beyond the glossary entry, for series of descriptions because there are already so many container elements. Therefore, an extra processing burden is assumed because lookahead is required to determine if there are multiple descriptions that have to be manipulated – for example, numbered.

Glossary lists are allowed in text as well as being mandated in the document glossary; lists in text would probably be manually created, although the document glossary is likely to be augmented.

5.1.4 Notes

The tree diagram in Figure 5-7 shows all the elements in the note class.

Figure 5-7: Notes



A **note** contains information that is especially important for the reader to see. A **caution** contains important information that describes possible harm to data or software. A **warning** contains important information that describes possible harm to hardware. A **danger** contains important information that describes possible harm to people.

An **editorial note** contains information that is important for readers of the document in its draft stages to see. It is assumed that when a document's draft status is not final, any editorial notes will appear, and when the document is in final form, formatting applications will suppress them from appearing in the output.

A note and an editorial note have the same content model. They contain an optional **title** followed by one or more elements from an IU5 mixture. If a title is not supplied, it is assumed that formatting applications will generate one.

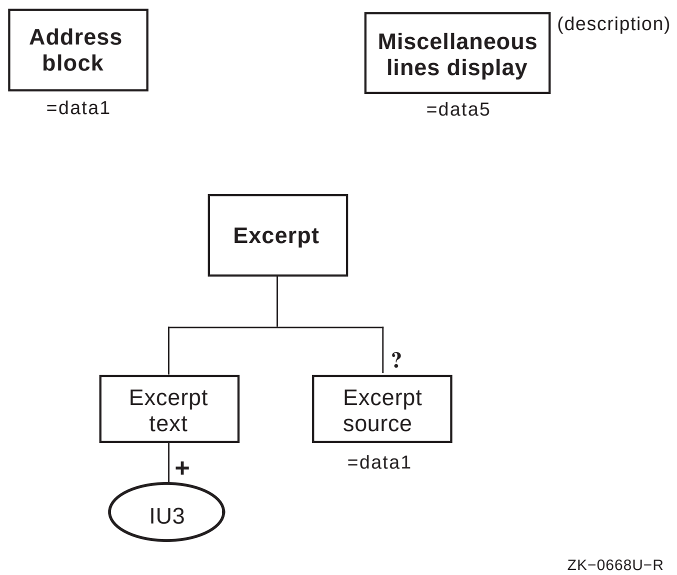
The caution, warning, and danger have the same content model: they contain one or more elements from an IU5 mixture. It is assumed that formatting applications will generate an appropriate title. Writers cannot supply a title on these components because standard, sometimes legally mandated, titles are usually required for the note components that describe possible harm.

A **note collection** contains a series of notes that are on a related subject. Typically a formatting application numbers the notes and generates a standard title for the collection, such as ‘‘Notes’’. Titles should be excluded from a note collection because the collection should receive a single title.

5.1.5 Address Block

The tree diagram in Figure 5-8 shows the address block.

Figure 5-8: Address Block, Miscellaneous Lines Display, and Excerpt



An **address block** is a displayed address; it is a specialized region that expects formatting applications to retain its record ends in the output. It contains a data1 mixture.

See Section 5.2.3 for information on the data-level address element.

5.1.6 Miscellaneous Lines Display

The tree diagram in Figure 5-8 shows the miscellaneous lines display.

A **miscellaneous lines display** is a displayed region that expects formatting applications to retain its record ends in the output. It contains a data5 mixture. It has a required **description** attribute to identify its contents.

5.1.7 Excerpt

The tree diagram in Figure 5-8 shows the excerpt.

An **excerpt** is an extensive displayed quotation from another source. It contains **excerpt text** and optional **excerpt source** information.

The excerpt text contains one or more elements from the IU3 mixture; this limited mixture is to accommodate some limited structure, such as paragraphs and lists, and is a moderate approach. Alternatives at the extremes would have been as follows:

- Allowing the exact simulation of the appearance of the original document, in which case the information would be more appropriately marked up as a figure
- Allowing the exact duplication of the structure and semantics of the original document, in which case a pointer to the document would be more appropriate

The availability of paragraphs and generalized lists should suffice for reproducing a limited amount of text from another document.

The excerpt source contains a data1 mixture. If the document name is mentioned, the key data element for document citation would presumably be used.

5.1.8 Models

The model class contains IUs that model the usage of various entities in computer languages, such as program routines and shell commands.

In using SGML to express characteristics of other computer languages, we have avoided trying to express every single feature of the other language. Doing so would be creating unnecessary semantic distinctions; it would also create a kind of SGML-form parser for the other language, and force writers to become ‘‘human compilers’’ just to code the material. While command models reach a sophisticated level of modeling sh: ft: not found

In addition, each kind of model is intended to encompass a wide range of languages; for example, the routine model should cover routines, procedures, and functions in many different programming languages. The more general each model is, the fewer features in any particular language it can be made to accommodate. If a need exists to model the features of a particular language more closely, a separate component for that model may be called for.

To determine the important distinctions, we examined the models and technical key data in the submissions. The technical key data reveals the individual software entities that probably need to be identified inside the models.

Argument information makes up a large part of most models. The design group recommends that the argument list component not be explicitly attached to any models, so that writers can organize them in the flow where desired. Where arguments and their placeholders appear in these models, they should allow links to the argument list entry where they are described, which functions as a weak form of ‘‘annotation’’ for the arguments. See Section 5.1.3.1 for information on argument lists.

5.1.8.1 Routine Model

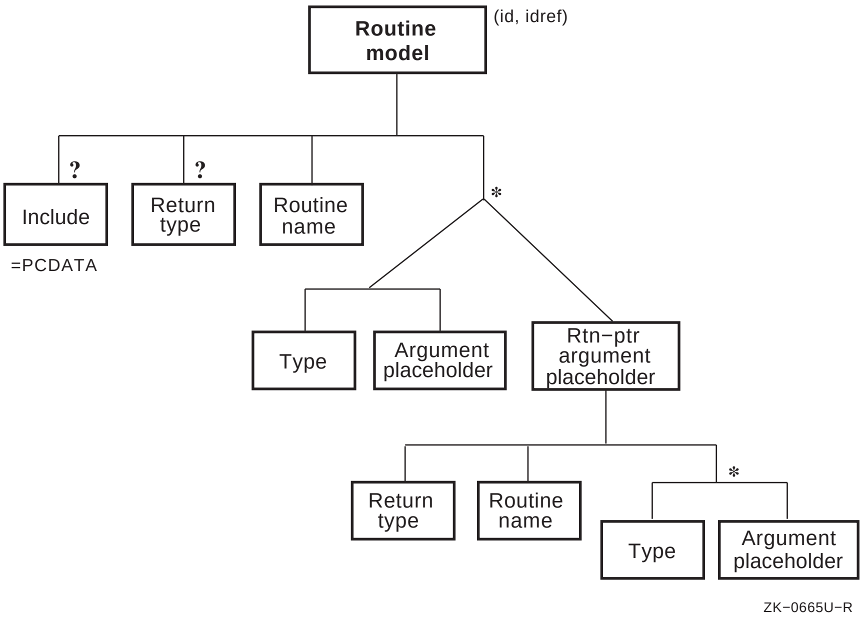
A **routine model** is a highly specialized component that describes the usage of a programming routine, procedure (different from the kind described in Section 5.1.14.2), or function. It has an **identification** attribute for reference from routine names mentioned in text.

The design group recommends at least the following components inside a routine model:

- The **routine name** (the same as the technical key data component)
- A **return data type**
- **Argument placeholders** and **argument data types**
- **Links** from the argument information to related routine models (such as when an argument points to another routine) and to argument list entries

The tree diagram in Figure 5-9 shows a possible organization for a routine model. However, further study and testing are needed in this area to ensure that the needs of the interchange partners are all accommodated (such as the issue of representing variable numbers of arguments, or of representing arguments with undefined data types).

Figure 5-9: Routine Model



5.1.8.2 Command Model

A **command model** is a highly specialized component that describes the usage of a command line, statement, directive, request, or other such software instruction. It is often known as the syntax or synopsis of the software entity.

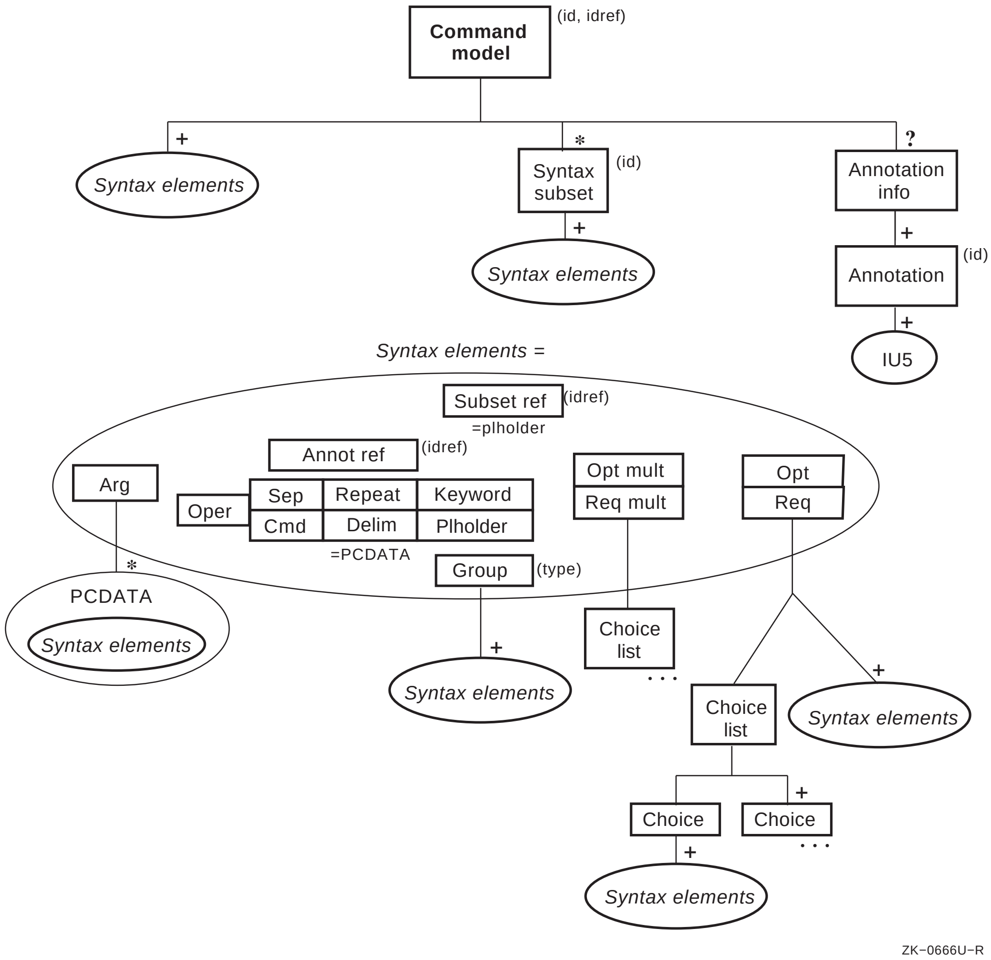
It has an **identification** attribute for reference from command names mentioned in text.

The design group recommends at least the following components inside a routine model. The main components described here are ‘‘syntax elements,’’ a mixture of which can appear in various contexts in a command model.

- Four choice operators that contain syntax elements, to indicate their occurrence options:
  - **Optional**, where some syntax element can be present or absent, or a single element can be chosen from among a list
  - **Required**, where some syntax element must be present, or a single element must be chosen from among a list
  - **Multiple-optional**, where zero or more syntax elements can be chosen from a list
  - **Multiple-required**, where one or more syntax elements must be chosen from a list
- A **choice list** with two or more **choice items** in it, for use inside the choice operators
- A **repeat operator** for syntax elements, such as arguments, that can repeat
- A **group** associating syntax elements that are distinct, but appear together in an unbroken block
- A command **argument** (the same as the technical key data element), which contains a mixture of PCDATA and syntax elements, and allows **links** to an argument list entry
- A **syntax subset reference** that provides a placeholder for a **syntax subset** for the model, and links to it
- An **annotation reference** that links to **annotations** for the model
- Simple syntax elements, which contain PCDATA:
  - **Command name** for the name of the command being documented (the same as the technical key data element)
  - **Keyword name** for the names of other software entities, such as options (the same as the technical key data element)
  - **Placeholder** for variable values that the user or the software must fill in (the same as the technical key data element)
  - **Separator** for separator characters, such as commas between file names in an argument
  - **Delimiter** for delimiter characters surrounding syntax elements, such as parentheses
  - **Operator** for operator characters, such as mathematical operators

The tree diagram in Figure 5-10 shows a possible organization for a command model. However, further study and testing are needed in this area to accommodate known needs, such as to indicate POSIX command options.

Figure 5-10: Command Model



The tree diagram shows first a mixture of one or more syntax elements as described above, followed optionally by one or more **syntax subsets**, followed optionally by an **annotation information** container.

A syntax subset is a logical portion of the command model, typically a very complex portion, that is supplied separately in the source from the main part of the model. The syntax subset reference (one of the syntax elements) assigns it a descriptive name and links to it; the name can be used as a placeholder if the subset is presented separately from the main portion of the model, which is typically done in hardcopy.

The annotation information is the same as the component that appears in some of the illustrations. The annotation reference (one of the syntax elements) links to the relevant annotation.

5.1.8.3 File Model

A **file model** is a highly specialized component that describes the usage of a file that has a certain structure, such as an initialization or build file.

A file model assumes that formatting applications will retain the record ends on output. However, it can probably have the same content model as a command model. The tree diagram in Figure 5-10 shows that model.

5.1.8.4 Data Structure Model

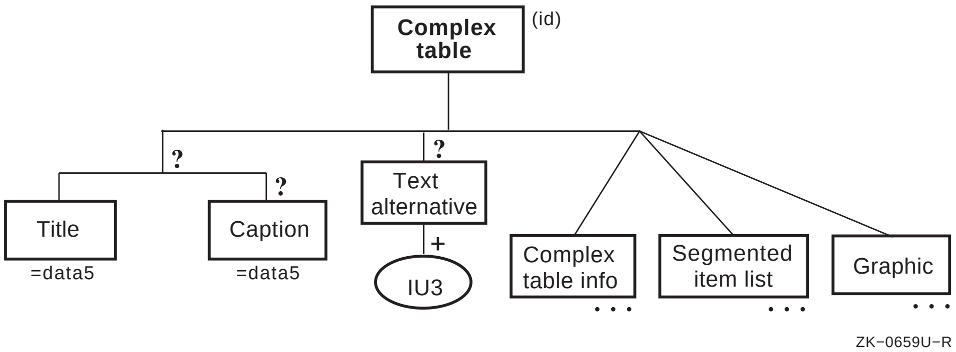
A **data structure model** is a highly specialized component that defines a data structure.

The design group recommends this component because of the value of identifying it uniquely among other models and computer dialog/states. However, our only specific recommendation for components inside it is to allow **links** to the models of routines that are pointed to from the structure. Further study is needed.

5.1.9 Table

The tree diagram in Figure 5-11 shows the table.

Figure 5-11: Table



A **table** is a multidimensional arrangement of information. It has an **identification** attribute for reference from elsewhere, and contains the following:

- An optional **title**, which if present, allows an optional **caption**
- An optional **text alternative** to the table that contains one or more elements from an IU3 mixture
- One of the following:
  - **Complex table** information encoded in SGML

    This component will probably need attributes that control the presentation of the table.
  - Table information encoded as a **segmented item list** (SIL, described in Section 5.1.2.4)

    SILs are allowed inside tables so that a writer can mark certain SILs as being tables in a logical sense; such SILs are expected to be presented to the reader as tables (for example, being formatted tabularly, and if they are titled, possibly being numbered and so on).
  - A **graphic** containing table information (see Section 5.2.2 for more information about graphics)

• An optional **key** explaining conventions used in the table

SILs, whether in tables or not, are by their nature only two-dimensional. Complex tables are typically more than two-dimensional, which in practical terms means they have information that needs to span cells either vertically or horizontally. The complex table component should have a superset of SIL capability, so that any material that SILs do not accommodate can use the complex table encoding.

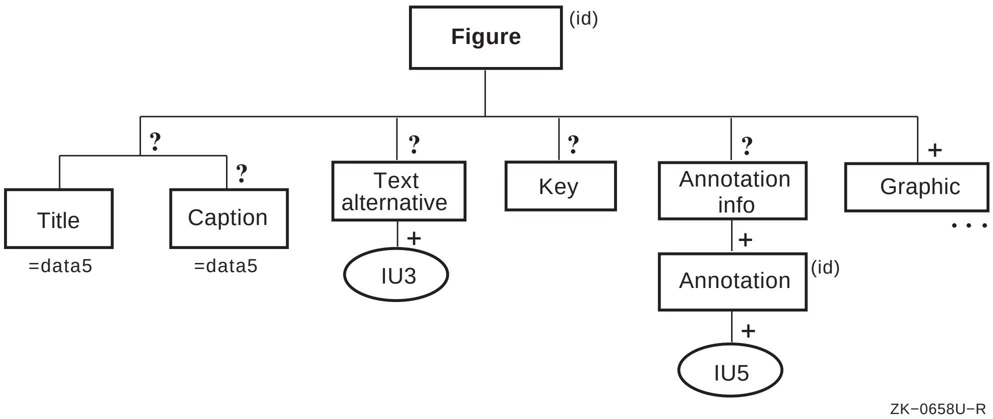
We have not developed an SGML encoding for complex table information. The encoding chosen should not force the selection of a single proprietary application as a table-formatting tool. Part 11 of ISO Technical Report 9573 (available from the GCA) offers a base tag vocabulary for tables that could be used as a starting point for the development of a complex table encoding.

The data mixtures in components inside the table might contain footnotes, which a hardcopy formatting application would typically number differently from footnotes outside table contexts and collect at the bottom of the table on output.

5.1.10 Figure

The tree diagram in Figure 5-12 shows the figure.

Figure 5-12: Figure



A **figure** is an illustration that is intended to be viewed as a picture. It has an **identification** attribute for reference from elsewhere, and contains the following:

• An optional **title**, which if present, allows an optional **caption**

• An optional **text alternative** to the figure that contains one or more elements from an IU3 mixture

• An optional **key** explaining conventions used in the figure

  The key is different from the annotations, which explain the real significance of points of the figure rather than merely explaining its pictorial vocabulary.

• An optional **annotation information** element, containing one or more **annotations** describing certain points in the figure

  Because figures are typically bitmaps, they probably cannot contain annotation references to link to the relevant annotation. However, the annotation has an **identification** attribute in case they can. The annotation contains one or more elements from an IU5 mixture.

- One or more **graphics**

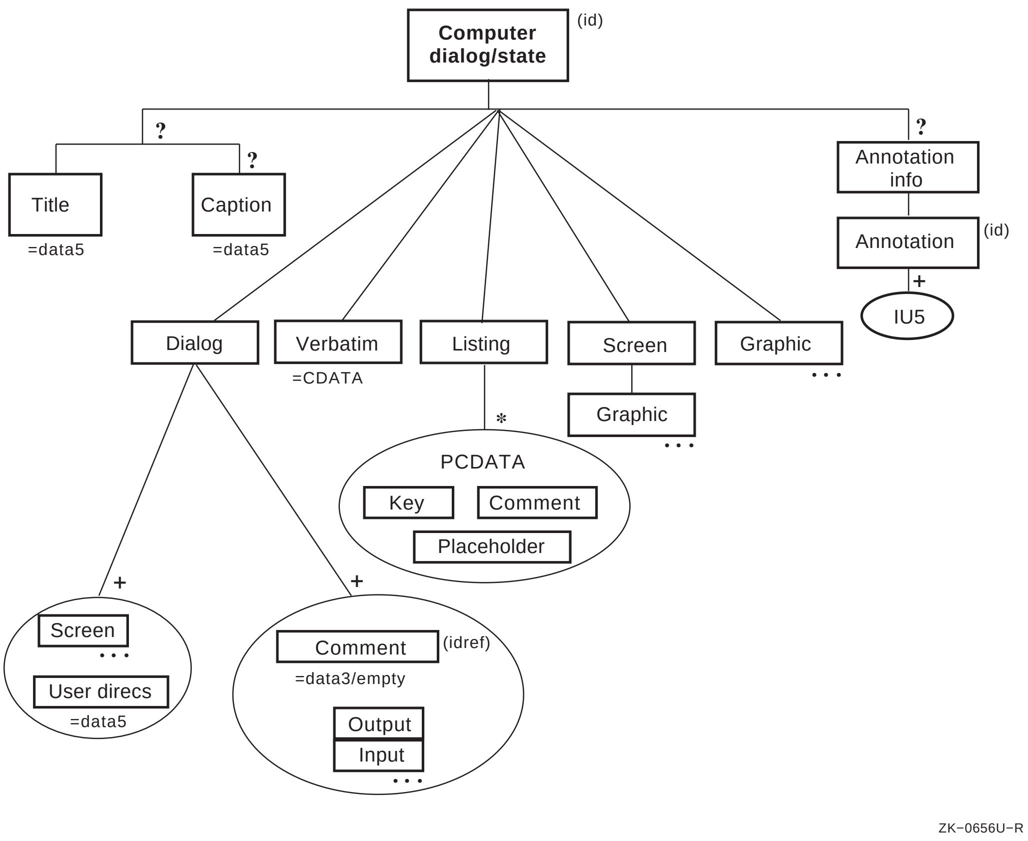
  A series is allowed when a series of graphics makes up the figure. The graphic has multiple entity attributes for the case when multiple versions of a graphic have been supplied in different notations.

  The graphic element is used in many places, such as other illustrations and within data-level mixtures. See Section 5.2.2 for more information.

5.1.11 Computer Dialog/State

The tree diagram in Figure 5-13 shows the computer dialog/state.

Figure 5-13: Computer Dialog/State



A **computer dialog/state** (CDS) contains an example of an aspect of computer software, for example, the state of the screen or an interaction between system and user. It is often called an example, exhibit, or listing in computer documentation. It has an **identification** attribute for reference from elsewhere, and contains the following:

- An optional **title**, which if present, allows an optional **caption**
- One of the following:

- A **dialog** demonstrating an interactive session that a user has with a computer

  It contains either a mixture of one more **screens** and **user directions** for describing interactions in a graphical environment, or a mixture of one or more descriptions of text-based user **input** and text-based system **output** interspersed with **CDS comments** for describing interactions in a keyboarding environment.

  The input and output elements are the same as the technical key data elements.

- A **verbatim listing** of program code or the contents of a file or buffer

  It contains CDATA so that characters that are problematic for SGML can be escaped.

- A regular **listing** of program code or the contents of a file or buffer

  It contains a mixture of PCDATA, **placeholders**, **keyboard key** data (the same as the technical key data), and **CDS comments**.

  If desired, the listing could have an entity attribute allowing the inclusion of a real program file, with the language used – C, FORTRAN, and so on – indicated as a non-SGML notation. The listing could then be extracted, tested, and otherwise manipulated by applications, including compilers.

- A **screen** graphic representing the state of the screen at any one point

  It has the same attributes as a graphic (see Section 5.2.2), and is empty.

- A **graphic** containing CDS information that must be physically represented as a graphic

  Graphics are described in Section 5.2.2.

• An optional **annotation information** element, containing one or more **annotations** describing certain points in the CDS. The annotation contains one or more elements from an IU5 mixture, and has an **identification** attribute for reference from a CDS comment.

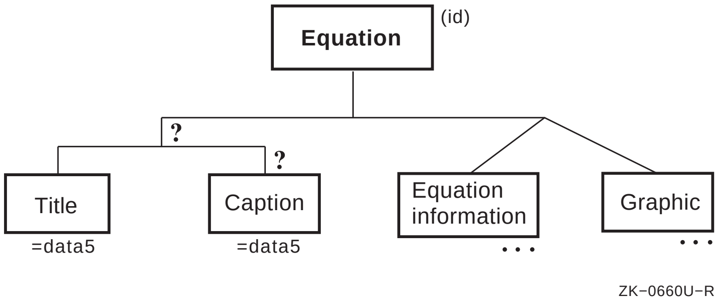
Listings, verbatim listings, and dialogs that contain text-based user input and system output assume that formatting applications will retain the record ends in the document output.

The CDS comments in dialogs and listings are for providing descriptions of certain points in the CDS. They can either contain a data3 mixture or link to an annotation provided with the CDS. In this way, extensive comments can be pointed to.

5.1.12 Equation

The tree diagram in Figure 5-14 shows the equation.

Figure 5-14: Equation



An **equation** is a mathematical equation or formula. It has an **identification** attribute for reference from elsewhere, and contains the following:

- An optional **title**, which if present, allows an optional **caption**

  Note that equations are often numbered, even if no title is present.

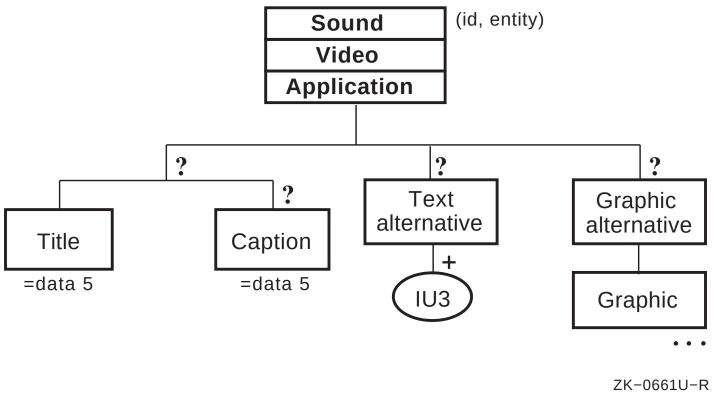
- One of the following:
  - **Equation information** encoded in SGML
  - A **graphic** containing equation information (see Section 5.2.2 for more information about graphics)

We have not developed an SGML encoding for equation information. The encoding chosen should not force the selection of a single proprietary application as an equation-formatting tool. Part 11 of ISO Technical Report 9573 (available from the GCA) offers a base tag vocabulary for equations that could be used as a starting point for the development of an equation encoding.

5.1.13 Dynamic Illustrations

The tree diagram in Figure 5-15 shows the dynamic illustrations.

Figure 5-15: Dynamic Illustrations



The **sound**, **video**, and **application launch** components all have an **identification** attribute for reference from elsewhere and an **entity** attribute indicating the location of the file that contains the illustration data, and have the same contents:

- An optional **title**, which if present, allows an optional **caption**
- An optional **text alternative** to the illustration that contains one or more elements from an IU3 mixture
- An optional **graphic alternative** to the illustration, which contains a **graphic** (described in Section 5.2.2)

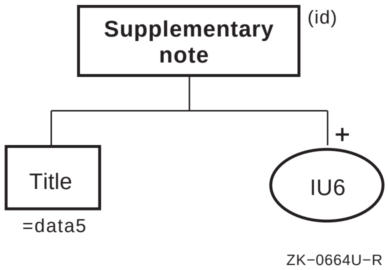
5.1.14 Division-Like Elements

The division-like elements are supplementary notes, procedures, and scenarios.

5.1.14.1 Supplementary Note

The tree diagram in Figure 5-16 shows the supplementary note.

Figure 5-16: Supplementary Note



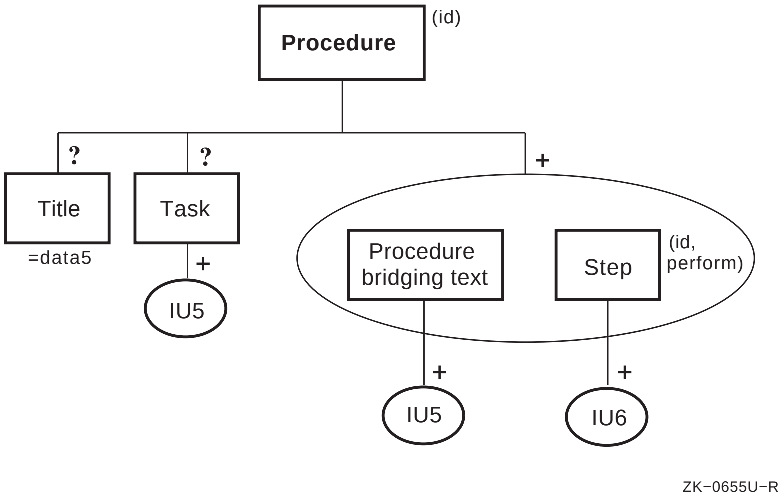
A **supplementary note** is an extended description of a topic that is peripheral to the discussion at hand, but is small or relevant enough not to be in its own appendix. It is similar to a boxed article or sidebar in a magazine. It belongs to the division that contains it, but the assumption is that processing will indicate that it is optional reading (for example, in hardcopy, boxing it and possibly floating it to a different location, or online, requiring a reader to click on something to see the material).

The supplementary note has a required **title** and one or more elements from an IU6 mixture. It has an **identification** attribute for reference from elsewhere (for example, a cross-reference from the text of the division).

5.1.14.2 Procedure

The tree diagram in Figure 5-17 shows the procedure.

Figure 5-17: Procedure



A **procedure** is a description of a task and the actions that are involved in performing that task. It contains the following, in order:

- An optional **title**
- An optional **task** description
- A collection of information on how to perform the task, consisting of any number of **steps** freely intermixed with any number of **procedure bridging text**, which contains a mixture of one or more elements from the IU5 mixture

  At least one element must be present here.

A step has an **identification** attribute for reference from elsewhere (for example, cross-references from the text), and a **performance** attribute that indicates whether the step is optional or required. This latter attribute is useful if different presentational emphasis or numbering is desired for optional versus required steps.

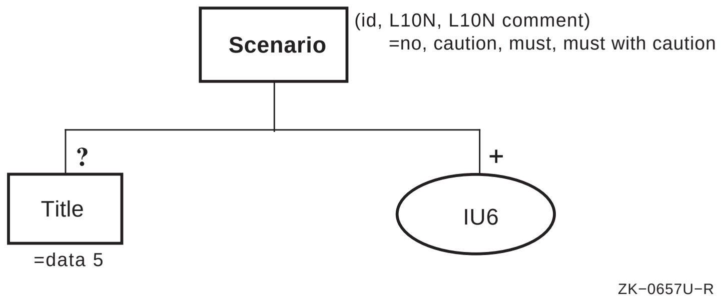
Steps do not have to be adjacent to each other. A procedure can be thought of as a recipe; like a recipe, it may need to supply warnings and supplemental information along with the steps to give more coherence to the instructions. There may not even be any steps as such, just a brief description of a simple action the reader must take. It is assumed that steps, if present, will probably be numbered in some way by an application.

Steps can contain nested procedures.

5.1.14.3 Scenario

The tree diagram in Figure 5-18 shows the scenario.

Figure 5-18: Scenario



A scenario is an illustrative example that may include computer dialog/states, text, or whatever other forms of information are needed to explain the circumstances of some aspect of the software.

For instance, there may be several ways to change a complicated system setup. A scenario can illustrate several ways that a person might perform the changes, and suggest which ways are best for which results. Encoding this description as a scenario allows it to be extracted and tested with all the relevant information present, as well as allowing it to be presented to the user as a whole package.

A scenario contains an optional **title** followed by one or more elements from an IU6 mixture. This mixture allows bridging titles to help structure the examples. The scenario has an **identification** attribute for reference from elsewhere, and the attributes **localization status** and **localization comments**, which are described in Section 2.2.10.

Typically, scenarios are given a generated title such as ‘‘Example’’. If a descriptive title is supplied by the writer, the title that is output could be something like ‘‘Example: Changing the System Setup’’. Scenarios could also be numbered and listed in a table of contents.

5.1.15 Bridging Title

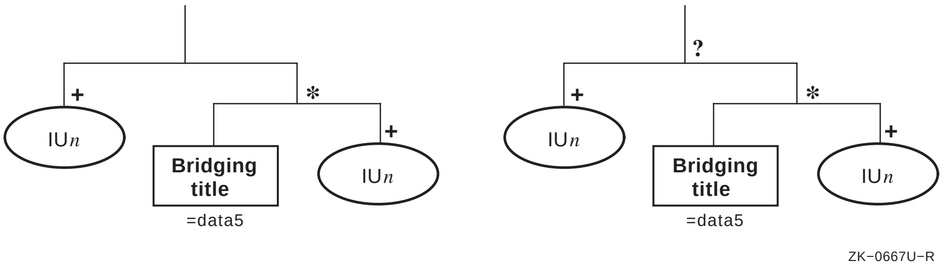
A bridging title is a title for a location within a division or division-like element. (See Section 2.1.2 for a fuller description of bridging titles.)

The IU1, IU3, and IU6 information unit mixtures all contain bridging titles. However, bridging titles are not available in a completely free mixture with the other IUs. A bridging title cannot appear first before any other information units; it cannot appear in series without other intervening information units; and it cannot appear last after any other information units.

Restricting the appearance of bridging titles in this way ensures that they cannot be used nonsensically, and also aids writers in using them properly. However, prototyping revealed that the special content models required for bridging titles may complicate the management of information pool entities and may contribute to some greatly increased capacities and quantities for the DTD. If basic restrictions on using the SGML language make this arrangement impossible, bridging titles may need to be allowed in a free mixture with other IUs.

The tree diagrams in Figure 5-19 show the bridging title and its relation to the IU1, IU3, and IU6 mixtures in which it is found. When one of these mixtures is mentioned in the recommendation text or in a tree diagram, the diagram shown here on the left is for where zero or more elements occur, and the one on the right is for where one or more elements occur.

Figure 5-19: Bridging Title and Its Contexts



5.2 Data

Data-level elements mostly have flat contents. Therefore, the following sections use only text, rather than tree diagrams, to describe the content models.

Note that many of the data-level elements are incompletely specified. Where we have recommendations on specific ones, these are noted.

5.2.1 Equation Data

The **equation data** element is different from the IU-level equation illustration described in Section 5.1.12. The equation data should contain the **equation information** encoding that is used for equation illustrations.

5.2.2 Graphic Data

Graphic data is available in several IU contexts; when it is used in one of the data mixtures as a peer to PCDATA, then it is assumed to be an inline graphic. These cases are usually small icons or graphical portions of trademarked terms. (If a keyboard key is represented with a graphic, technical key data element for keyboard keys should be used.)

The graphic element is empty. It has, at the least, several **entity** attributes that contain versions of the graphic in the different notations that have been approved for interchange. Each attribute should be associated by convention with a particular notation type. (Section 6.2.1 describes issues surrounding graphics notation recommendations.)

5.2.3 General Key Data

The general key data elements identify small pieces of information that are unrelated to specific aspects of software. The design group has tended to recommend all the components submitted.

Many of these elements should probably allow data mixtures because they contain real natural-language information; some may need to exclude themselves from the mixture.

The design group recommends the following general key data:

- **Acronym** for acronyms and abbreviations

  Marking up this kind of information allows applications to, for example, check it against a dictionary.

- **Document citation** for reference to outside documents

  This component is expected to be used in bibliography entries. This may serve as the same component that **links** to outside documents (mentioned in Section 5.2.5). It might also allow **links** to a bibliography entry in this document.

- **Document author**

  This is especially important in bibliography entries, but is available generally.

- **Glossary term in text**

  This component allows writers to mark up terms in the text that have glossary entries. The component can be used selectively on the instances of a term where the writer considers it important to inform the reader about the glossary entry, or its use can be globally added by augmentation. It allows **links** to the entry or an individual definition within the entry. It is different from the component that contains the glossary term within the entry itself.

  The presence of instances of this component may also be the means by which a glossary is generated or augmented.

- **New term in text**

  This component is for terms that are newly introduced and defined at this location in the text. It is different from the component for glossary terms in text, which do not necessarily mark up locations where the term is defined.

  For manuals that have glossaries, good style dictates that every new term defined in the text should also have a glossary entry. Such terms in text can be marked up simultaneously with both the glossary term and the new term elements.

- **Emphasized phrase**

  This component is for emphasizing or highlighting a word or phrase. It should not allow specific typographical control.

  If it is important to distinguish between an emphasized phrase that draws the reader's attention to its meaning and a highlighted string that draws the reader's attention to its presence or location, an attribute can be used to make the distinction.

- **Phrase in a different language**

  Phrases marked up with this component can be treated specially for spellchecking and translation purposes, as well as possibly receiving typographic emphasis. The language should be indicated in an attribute.

- **Localization phrase**

  This component might be able to be combined with the previous one. Phrases marked up with this component need special localization attention. It has two localization attributes, which are described in Section 2.2.10.

- **Quotation**

  This component is different from the excerpt IU. Formatting applications are expected to produce matching quotation marks of the appropriate type around the phrase. (Entities should be used to produce single instances of quotation mark symbols.)

- **Subscript**
- **Superscript**
- **Trademarked term in text**

  This component allows **links** to the trademark attributions in the metainformation. Formatting applications can use this component to produce the proper TM, R, or SM symbol on the term in text, to produce attribution footnotes, to verify that all the necessary attributions have been provided in the metainformation, or to build the attribution list by augmentation.

- **Address data**

  This component is different from the address block IU. However, it has the same content model; see Section 5.1.5.

- **Date**

  This component is needed for localization. It allows PCDATA inside it, and has required number-value attributes for the **month**, **day of month**, and **year**.

  The attributes are to satisfy an OSF requirement to indicate these values for internationalization processing, although we expect that the supplied content will more often be used as it is supplied, or manipulated independently of the attributes in computer-aided translation. This solution unfortunately requires information to be repeated in two places. However, the other alternatives are unsatisfactory: Either the presence of month, day, and year elements cannot be ensured when they are peers to PCDATA needed for punctuation, or the month, day, and year elements would be the only contents of the data element and formatting applications would need to generate all the various kinds of date phrasings needed (‘‘Fourth of July’’, ‘‘4 July 1992’’, July 4, 1992’’, and so on) because PCDATA has been eliminated.

  We do not recommend including an explicit hook for the generation of a ‘‘system date,’’ either as an attribute to the date element or as a PI, because we do not want to encourage the presence of a dynamic ‘‘current date’’ in interchange. An explicit date should be inserted by the time the document is interchanged. However, the use of such a PI cannot be prevented.

- **Time**

  This component is needed for localization. We agreed that it is unnecessary to encode its internal structure.

- **Calendar reference**

This component is needed for localization. It marks up references to, for example, the ‘‘first day of the week’’ (which is different for different countries).

- **Telephone number**

  This component is needed for localization.

- **Proper name**

  This component is needed for localization of the names of people and places.

- A person’s **title**

  This component is needed for localization. It marks up a person’s title, degree, or honorific.

- **Unit of measure**

  This component is needed for localization. Rather than encode the particular unit used, this component serves as a specialized warning that tells translators to look carefully at this information.

- **Miscellaneous key data** with required **description** attribute

  This component is needed to account for unforeseen needs.

5.2.4 Technical Key Data

The technical key data elements identify small pieces of information that are related to specific aspects of software. The design group has tended to recommend most of the components submitted, although this may have increased the risk of chips.

Many of these elements should probably contain simply PCDATA; those that have more complex mixtures may need to exclude themselves from the mixture.

The design group recommends the following technical key data:

- **Literal string of characters**

  This component is for, for example, a string showing the acceptable characters in file names.

- **Numeric constant**

  OSF has indicated that it makes constant names look different from other technical key data, which means it must be distinct from other components.

- **Binary** and **hexadecimal** numbers

  These might be able to be a single component with an attribute indicating the radix. The attribute values could be chosen from a token list.

- **Logical negation**

  This hardware-related component is for descriptions of negative active signals.

- **Keyboard key**

  This component is for both real keycap names and ‘‘logical’’ keys, such as the name of a function performed by a function key. It is also available in various places in computer dialog/states. It contains a data mixture that allows graphics.

- **Graphic interface text**

  This component is for identifying text that is reproduced from a windowing interface, for example, menu item names. These are often made to look different from other technical key data.

- **Argument**

  This is also available in command models. It allows a mixture of components related to command models (including itself), along with PCDATA. This component can be used to provide limited inline syntax at the data level.

  It allows **links** to argument list entries.

  See Section 5.1.8.2 for more information.

- **Keyword**

  This component is for the formal names of software entities other than command and routines, such as options, environment variables, and various programming entities. It is also available in command models.

  It allows **links** to reference pages and command models.

- **Command name**

  This component is for the formal names of commands and other command-like software entities: statements, requests, directives, and instructions. It is also available in command models.

  It allows **links** to reference pages and command models.

- Text-based user **input**

  This is also available in computer dialogs. It should contain at least a mixture of PCDATA, keyboard keys, and placeholders.

- Text-based system **output**

  This is also available in computer dialogs. It can be used for fragments of programming code that are larger than a single keyword or routine name. It should contain at least a mixture of PCDATA, keyboard keys, and placeholders.

- **Placeholder**

  This component is to assign a mnemonic name to a variable value or series of values that a user or the system must supply. It is different from the general key data emphasis element.

- **Message code**

  This is also available in items in message list entries. It allows **links** to message list entries.

- **Message text**

  This is also available in items in message list entries. It allows **links** to message list entries.

- **Routine name**

  This component is for the formal names of routines and other similar software entities: procedures and functions. It is also available in routine models.

  This component is for the formal names of routines, procedures, and functions used in programming. It is also available in routine models.

  It allows **links** to reference pages and routine models.

If file and directory naming information needs to look distinct from the technical key data components described here, an element might be needed for it.

5.2.5 Cross-Reference Links

Cross-reference links provide a means to link to elements, locations, or documents, while ensuring that an application generates text describing those targets.

The design group recommends **cross-reference links** to the following:

- Locations (spots) within the current document or an external document

  This kind of cross-reference requires an element to associate locations in data with IDs: a **spot** element. Index entries in data should also be able to be cross-referenced as spots. See Section 5.2.7 for more information.

- List items
- Procedure steps
- Illustrations – tables, figures, computer dialog/states, equations, and dynamic illustrations – within the current document or an external document
- Divisions within the current document or an external document
- Supplementary notes
- External documents

  This is different from the general key data element for document citation.

- Reference pages

Any instance of one of these to which a cross-reference is made must have an ID. These IDs can also be used for generated links from tables of contents and indexes.

The assumption is that sensible text can be generated, using text such as illustration or division titles or location page numbers.

To link to external documents or things inside them, a level of indirection is needed because SGML mechanisms (such as ID/IDREF) cannot work across document instances.

5.2.6 Footnote Links

A **footnote** is essentially an annotation (like those found in illustrations) that is attached to some location in data. However, whereas annotations explain aspects of an illustration that are presentation sensitive, a footnote explains something in narrative text.

The design group concluded that table footnotes are like footnotes rather than annotations, because they explain a table's text rather than its presentation-sensitive aspects. We recommend that footnotes inside and outside tables be considered the same component and use the same element. Context can determine whether the footnote applies to table text.

Footnotes do two jobs: marking a location and supplying explanatory text for that location. Footnote components often refer to some already-supplied text rather than supplying it; for example, when the same footnote applies to two different locations, the first location supplies the text, and the second location points to the first one's text. Therefore, we concluded that a single footnote component should allow either explanatory text or a pointer to it.

(The same is true of annotations in computer dialog/states; some comment text can be supplied directly in the CDS, but sometimes extensive text cannot conveniently be embedded into the CDS. A single comment component should allow either comment text or a pointer to the annotation stored elsewhere. See Section 5.1.11 for more information.)

5.2.7 Other Data Links

The design group recommends that a **generalized link** component be provided to account for unforeseen needs. It should be able to surround data and, at the least, link to any of the things that a cross-reference can link to. While this is nominally a silent link since it generates no text that must be incorporated into a flow, the purpose of having this component would be to enable online linking facilities such as hotspots on which the user can click to go to the target.

In addition, because cross-reference and generalized links need to point to locations in data, a means is necessary to assign IDs to such locations. A **spot** element is needed to do this. In addition, **index entries** are needed at the data level to provide a means for links from an index to the document, as well as to serve as spots.

Links at the data level that are specific to certain semantic components, such as links from command names to reference pages, are described in the key data section where their host element is described; they are considered part of key data rather than part of this class of links.

Other Recommendations 6

This chapter describes recommendations for additional features of the DTD.

6.1 Document Revision Information

Identifying the new and changed information in documents is useful, not just for the insertion of change bars in presentation instances but also for other reasons, such as allocating translation resources, or stocking a document database.

Marked sections and asynchronous elements (two separate elements forming the ‘‘start tag’’ and ‘‘end tag’’ of a single semantic component) are typical solutions. The design group has not recommended a specific solution for marking up such information, but we observed that both of these solutions are problematic; they can interfere with a document’s tree structure by creating container boundaries for revised information at points other than synchronous element boundaries.

6.2 Notations

The following sections describe recommendations for notations.

6.2.1 Graphics Notations

The IAWG’s Graphics Notations Requirements paper outlines some basic requirements for graphics notations. The paper is a valuable resource for information on types of notations and their characteristics and the needs of OSF interchange partners. The OSF *DTD Documentation Tools Plan* discusses additional issues surrounding the choice of notation. However, the design group has made no specific recommendations for graphics notations, other than that a finite list of acceptable notations be chosen.

A CDATA or ‘‘flat ASCII file’’ notation might be useful for graphics that have been sketched with regular keyboard characters for display on character-cell devices.

6.2.2 Other Notations

Section 5.1.11 mentions the possibility of allowing listings to include real program files, indicating the computer language used in the file as a non-SGML notation. If this suggestion is used, whatever languages are appropriate for this treatment (certainly at least C) would need a notation declaration.

6.3 Entity Sets

Barring the use of validating applications that prohibit it, SGML document instances can include declarations for, and references to, any general entities that are desired. (Section 1.4.4 makes recommendations about restricting parameter entities added on an instance-specific basis; the same would apply to any restrictions on general entities.) However, there are some entity sets that should be available through the DTD to all interchange partners.

Entities are needed for any characters and symbols that cannot be entered directly into an instance with a keyboard. (Our recommendation for the unordered list also relies on the availability of entities for any symbols needed as bullet characters.) Various interchange partners have given OSF information on the symbols those companies use. We recommend that the DTD use ISO entity sets for any necessary sets of symbols.

Also, any case where text needs to be standardized in OSF interchange, or where information will be written once but used across many OSF-related documents, is a good candidate for a general entity available through the DTD. For example, our recommendations mention that the manual type component in the document ID metainformation will likely contain text from a standard list of manual type descriptions. For another example, trademark attributions use standard wording and would benefit by being defined in a general entity set and pulled into documents as necessary; a standard trademark attribution entity set would essentially form a trademark database.

Finally, the design group has uncovered several areas where processing instructions (PIs) may make sense, even though processing information is normally anathema to interchange. At the least, PIs in interchanged documents should use entities so as to allow receivers to negate or change their effects; it has been suggested that a small ‘‘approved’’ OSF PI entity set would help receivers deal with any PIs that are used in documents. The following possible PI entities are offered for consideration (although our only recommendation is that they not be implemented as semantic components):

- Hyphenation override in an instance of a word
- Keeping a word together on a line, indicating a soft (valid) line break in a word, and indicating a hard line break in a word
- Keeping information together on a page, indicating a soft (valid) page break, and indicating a hard page break
- Producing small capital letters (sometimes needed to simulate trademark appearance accurately)
- Indicating data-level material in a draft document that requires verification (for example, in an installation procedure, an estimated number for the amount of required disk space supplied temporarily until the precise number is known)

Tree Diagram Summary A

This appendix collects all the tree diagrams found in Chapter 3, Chapter 4, and Chapter 5 in one place for easy reference. The tree diagrams provide a useful overview of the recommendations, but the recommendations are explained completely only in the text.

The tree diagrams make reference to standard IU and data mixtures; see Table 2-1 and Table 2-2 for information on the contents of these mixtures.

Figure A-1 explains how to read the tree diagrams, and Figure A-2 shows examples of interpreting them in SGML.

Figure A-1: Key to Reading the Tree Diagrams

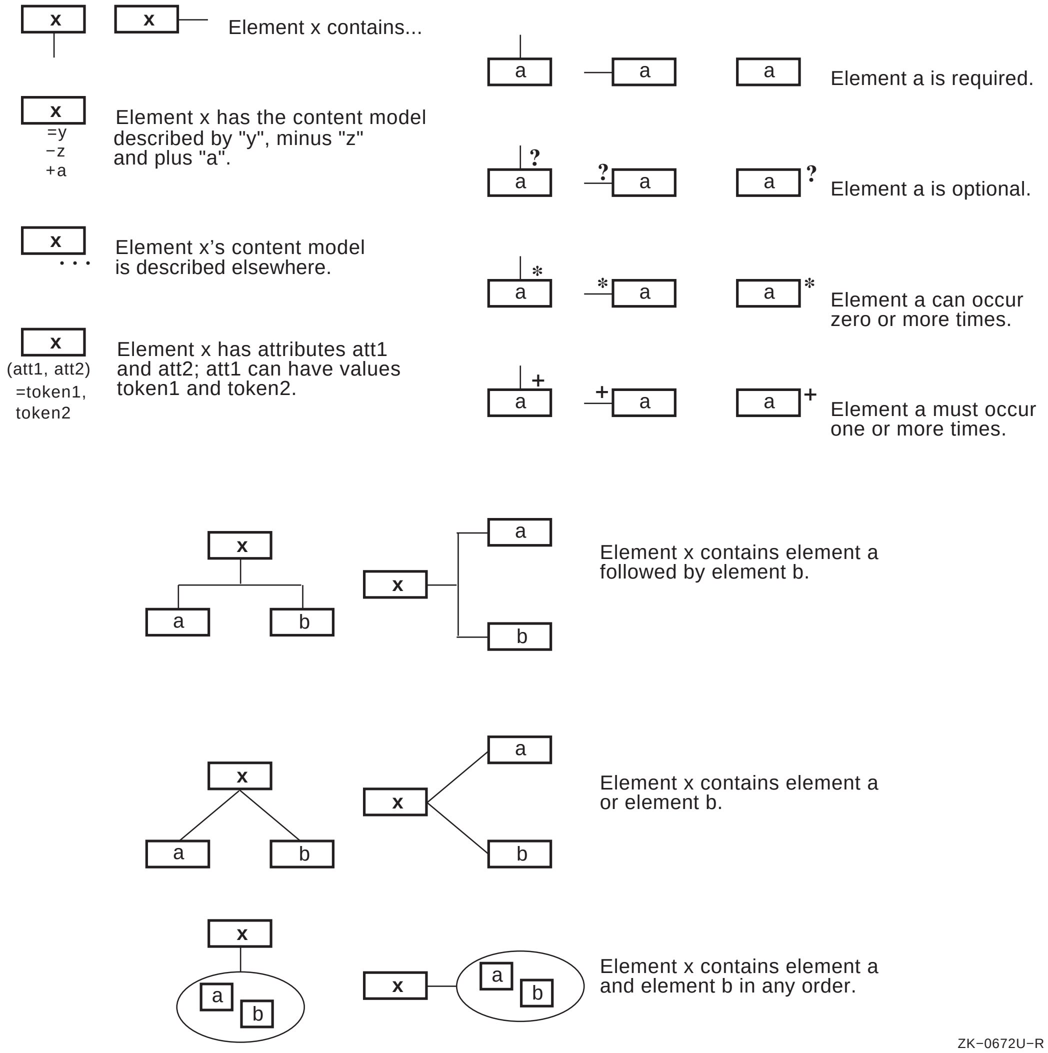
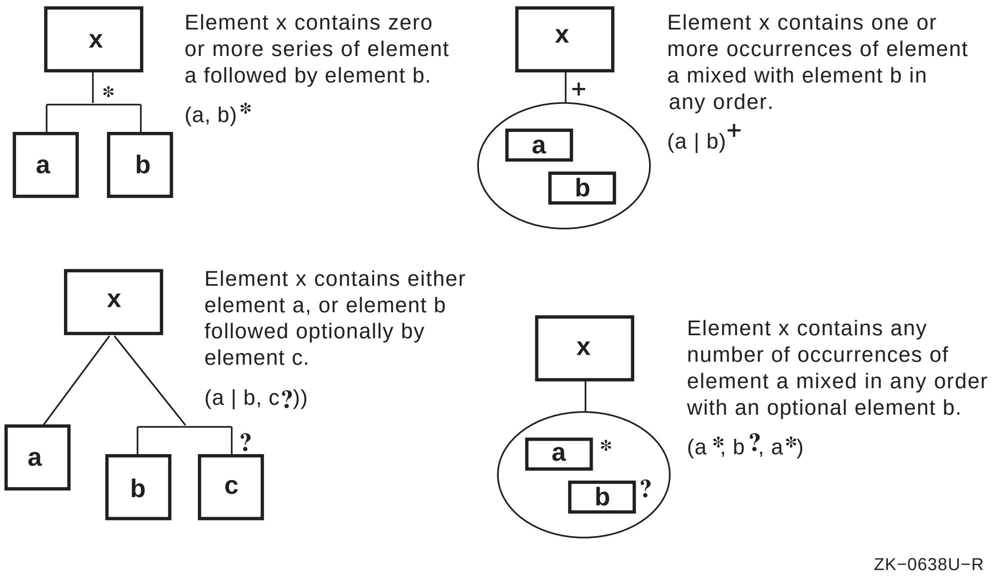
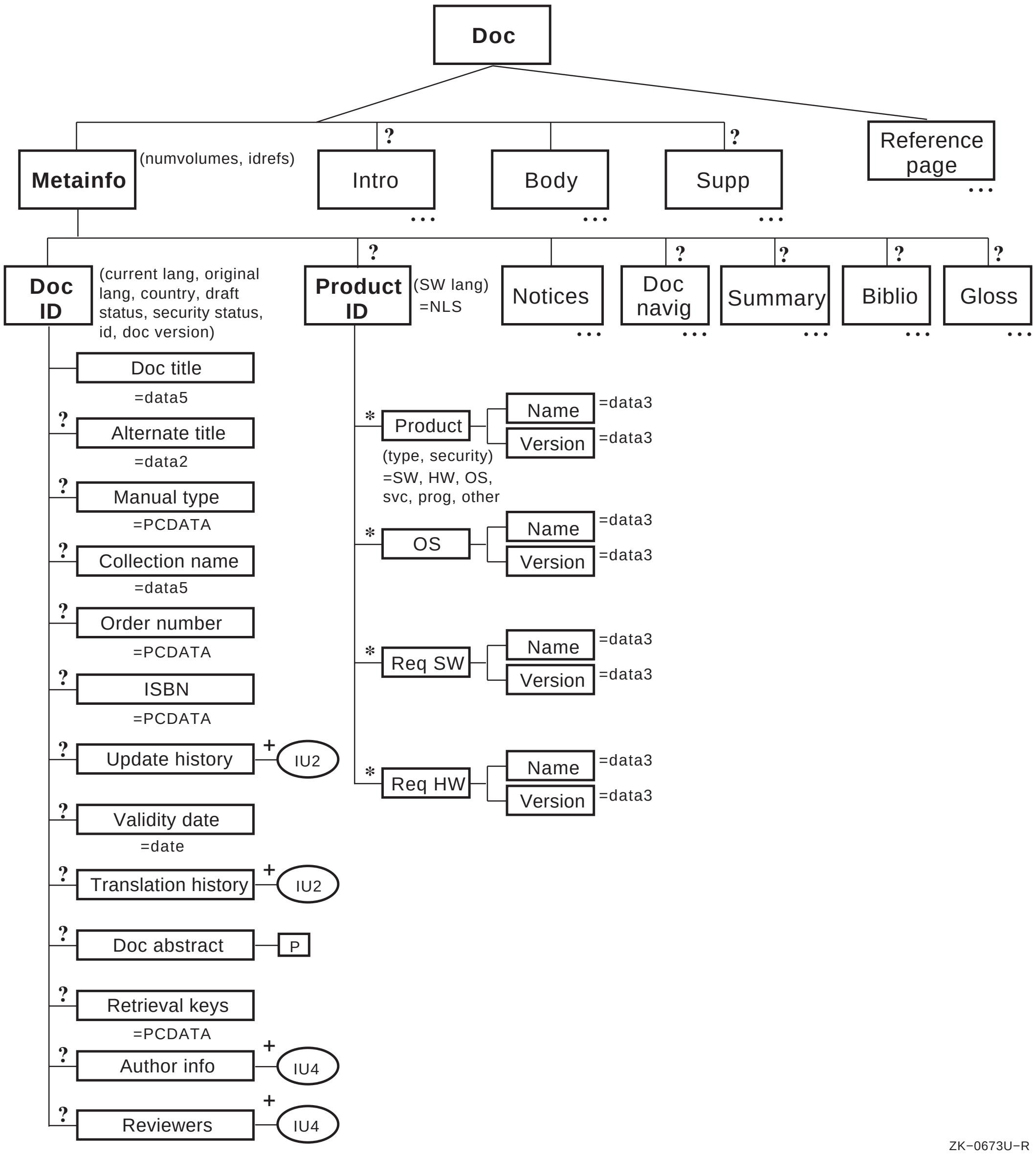


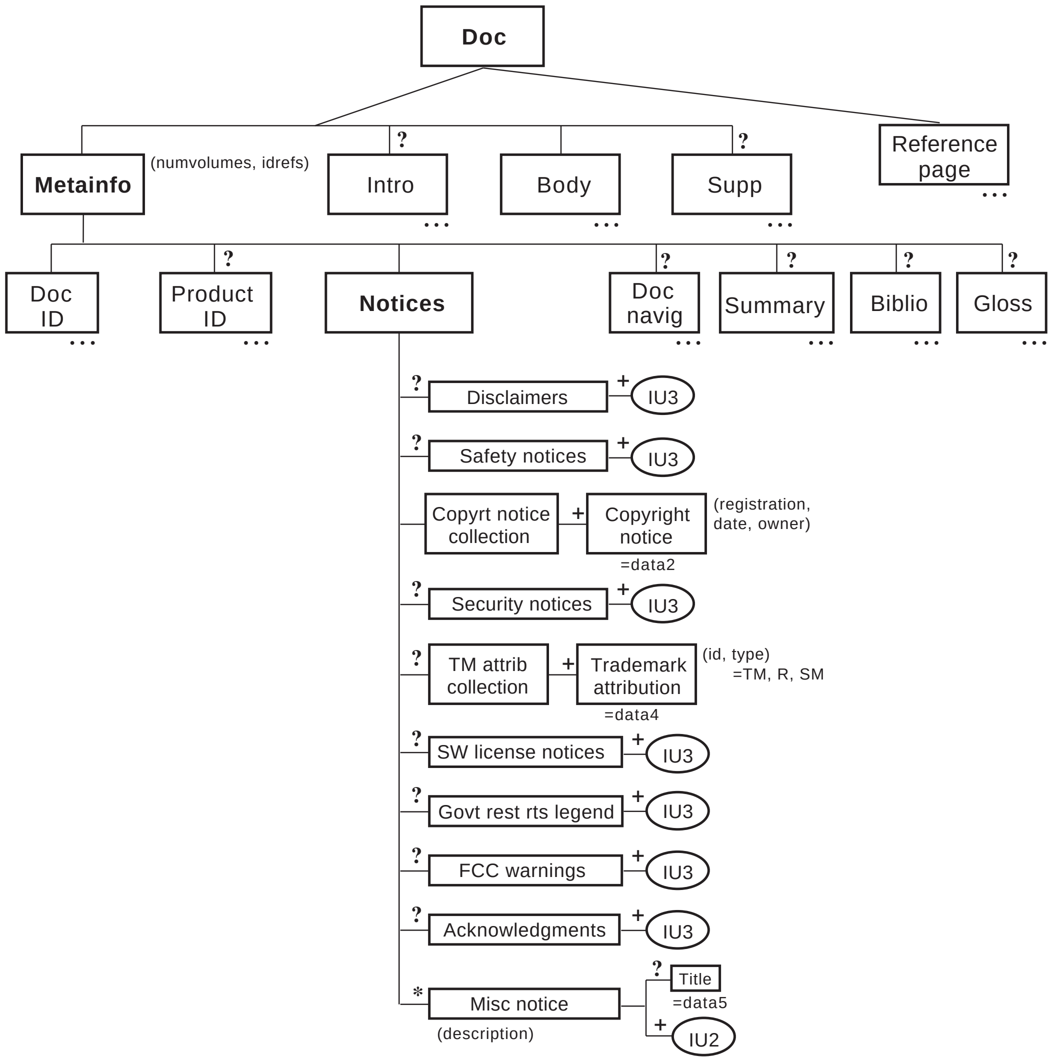
Figure A-2: Examples of Tree Diagrams

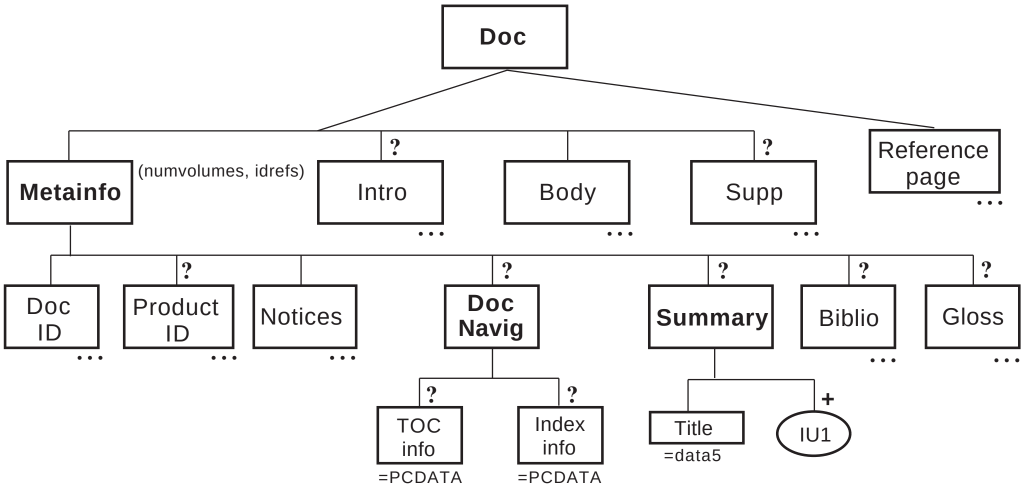


A.1 Tree Diagrams for the Manual Organization

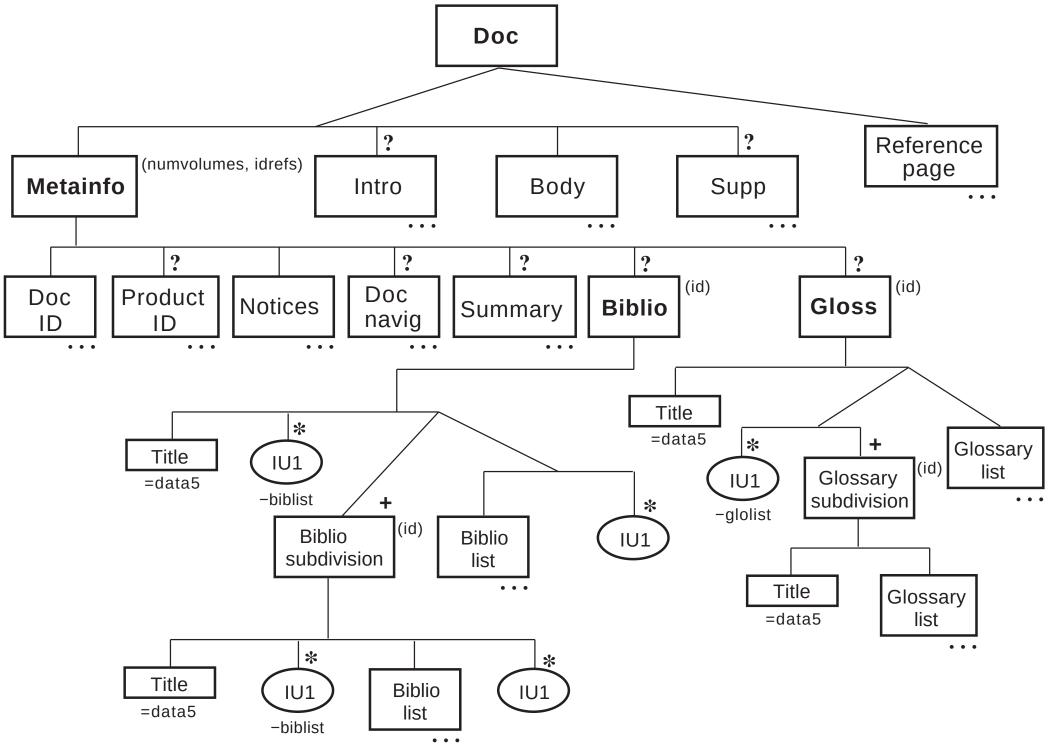
This section collects the tree diagrams for the manual document type organization.



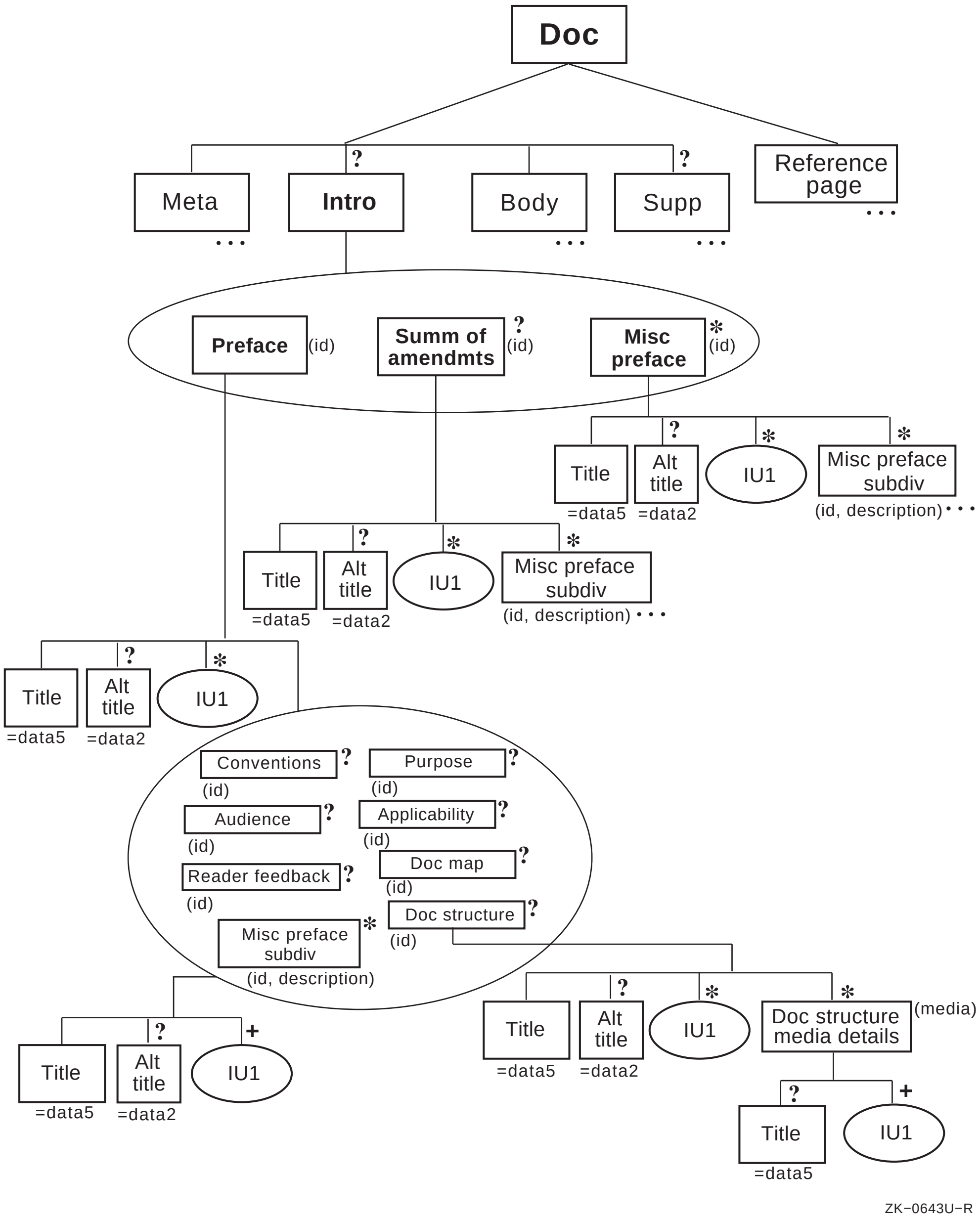

Doc
Metainfo
(numvolumes, idrefs)
?
Intro
Body
?
Supp
Reference page
Doc ID
?
Product ID
Notices
?
Doc navig
?
Summary
?
Biblio
?
Gloss
?
Disclaimers
+
IU3
?
Safety notices
+
IU3
Copyrt notice collection
+
Copyright notice
(registration, date, owner)
=data2
?
Security notices
+
IU3
?
TM attrib collection
+
Trademark attribution
(id, type)
=TM, R, SM
=data4
?
SW license notices
+
IU3
?
Govt rest rts legend
+
IU3
?
FCC warnings
+
IU3
?
Acknowledgments
+
IU3
*
Misc notice
(description)
?
Title
=data5
+
IU2
ZK−0674U−R



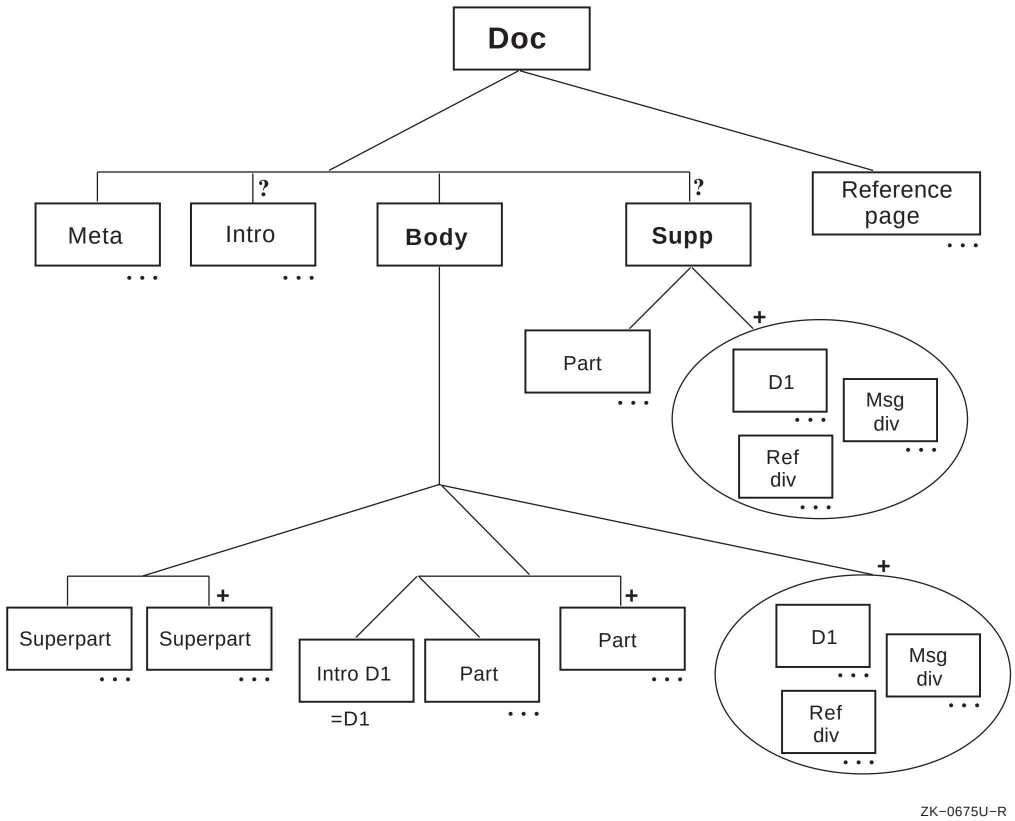
ZK−0639U−R

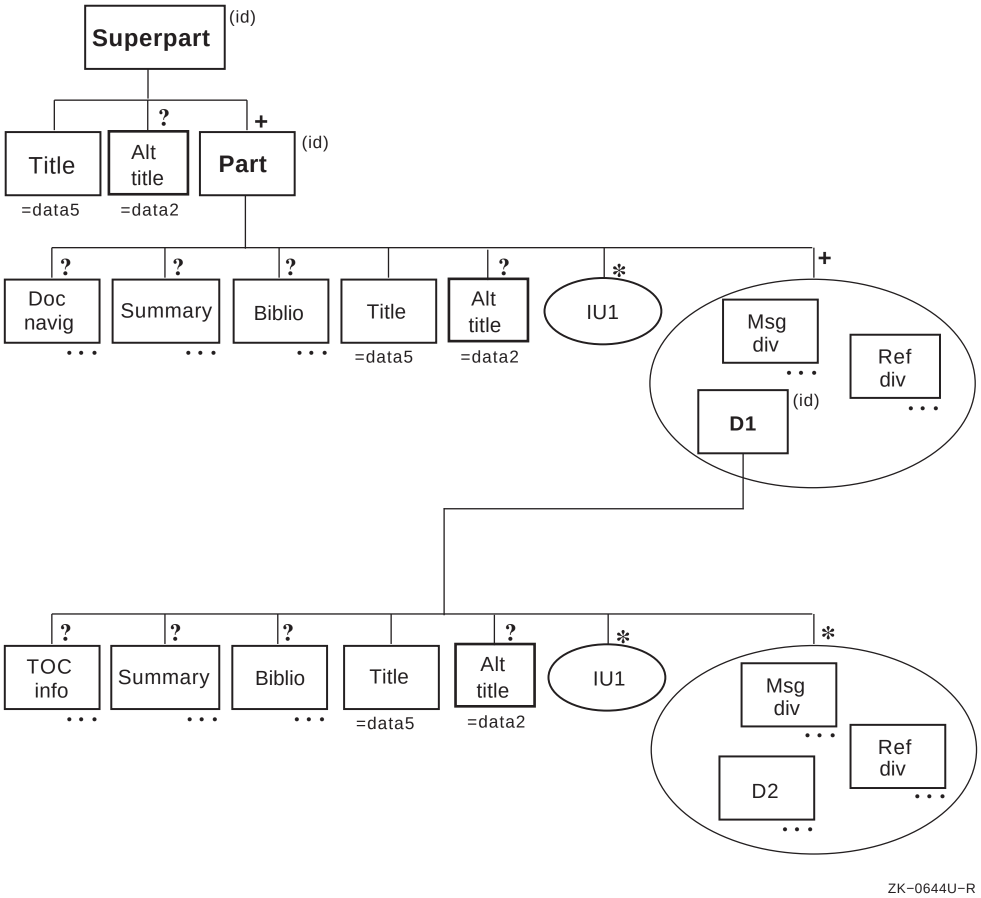


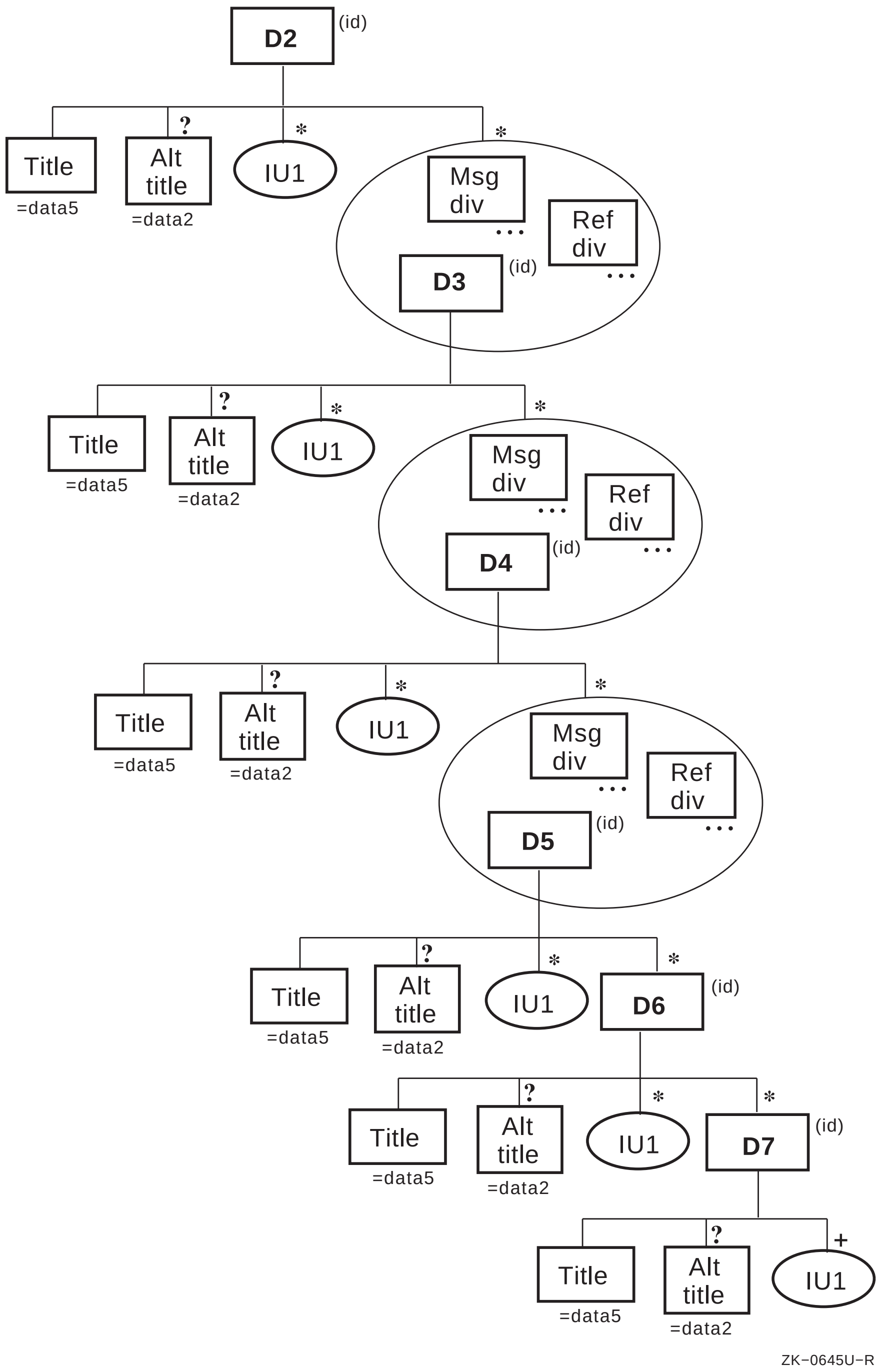
ZK−0642U−R

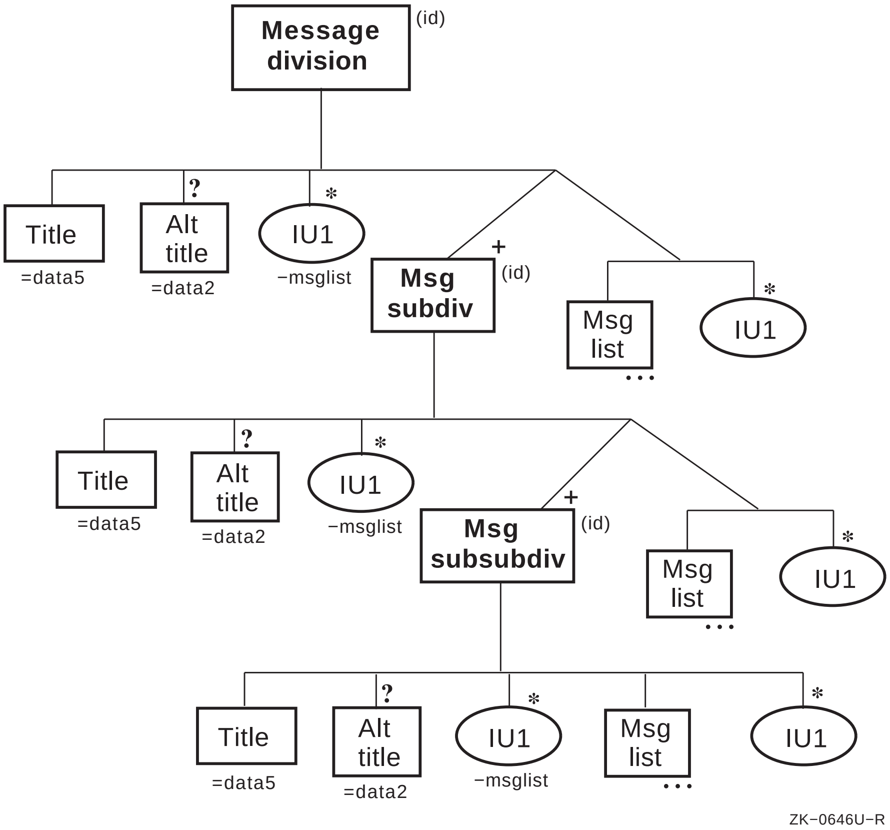


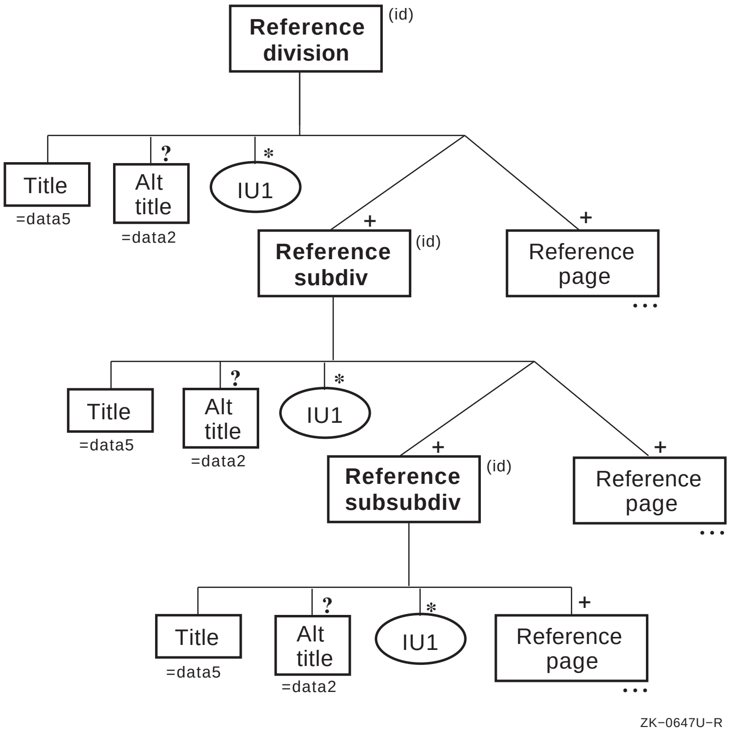
Doc
Meta
Intro
Body
Supp
Reference page
Preface
(id)
Summ of amendmts
(id)
Misc preface
(id)
Title
=data5
Alt title
=data2
IU1
Misc preface subdiv
(id, description)
Conventions
(id)
Purpose
(id)
Audience
(id)
Applicability
(id)
Reader feedback
(id)
Doc map
(id)
Doc structure
(id)
Misc preface subdiv
(id, description)
Doc structure media details
(media)
ZK−0643U−R


Doc
Meta
Intro
?
Body
Supp
?
Reference page
Part
+
D1
Msg div
Ref div
Superpart
Superpart
+
Intro D1
=D1
Part
Part
+
+
D1
Msg div
Ref div
ZK−0675U−R


Superpart
(id)
?
+
Title
=data5
Alt title
=data2
Part
(id)
?
?
?
?
*
+
Doc navig
Summary
Biblio
Title
=data5
Alt title
=data2
IU1
Msg div
Ref div
D1
(id)
?
?
?
?
*
*
TOC info
Summary
Biblio
Title
=data5
Alt title
=data2
IU1
Msg div
Ref div
D2
ZK−0644U−R

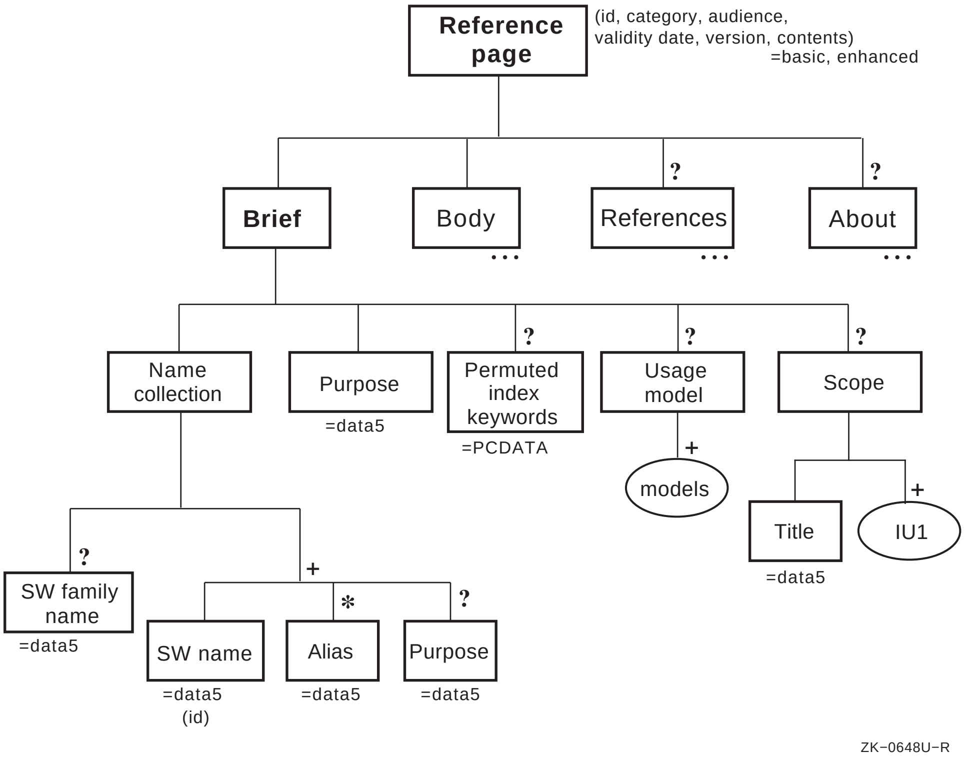

D2
(id)
?
*
*
Title
=data5
Alt title
=data2
IU1
Msg div
...
Ref div
...
D3
(id)
?
*
*
Title
=data5
Alt title
=data2
IU1
Msg div
...
Ref div
...
D4
(id)
?
*
*
Title
=data5
Alt title
=data2
IU1
Msg div
...
Ref div
...
D5
(id)
?
*
*
Title
=data5
Alt title
=data2
IU1
D6
(id)
?
*
*
Title
=data5
Alt title
=data2
IU1
D7
(id)
?
+
Title
=data5
Alt title
=data2
IU1
ZK−0645U−R


Message division
(id)
?
*
Title
=data5
Alt title
=data2
IU1
−msglist
+
Msg subdiv
(id)
*
Msg list
IU1
?
*
Title
=data5
Alt title
=data2
IU1
−msglist
+
Msg subsubdiv
(id)
*
Msg list
IU1
?
*
*
Title
=data5
Alt title
=data2
IU1
−msglist
Msg list
IU1
ZK−0646U−R


Reference division
(id)
Title
=data5
Alt title
=data2
IU1
Reference subdiv
(id)
Reference page
Title
=data5
Alt title
=data2
IU1
Reference subsubdiv
(id)
Reference page
Title
=data5
Alt title
=data2
IU1
Reference page
ZK−0647U−R

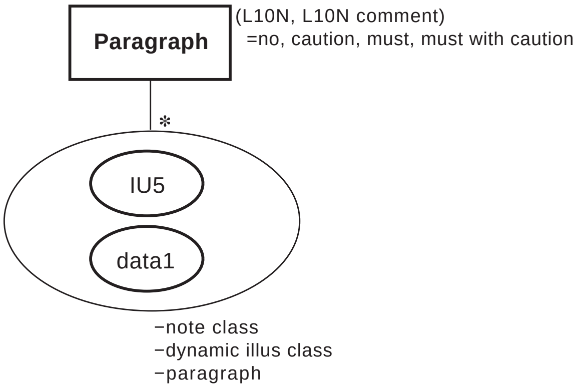
A.2 Tree Diagram for the Reference Page Organization

This section contains the tree diagram for the document type organization for reference pages.

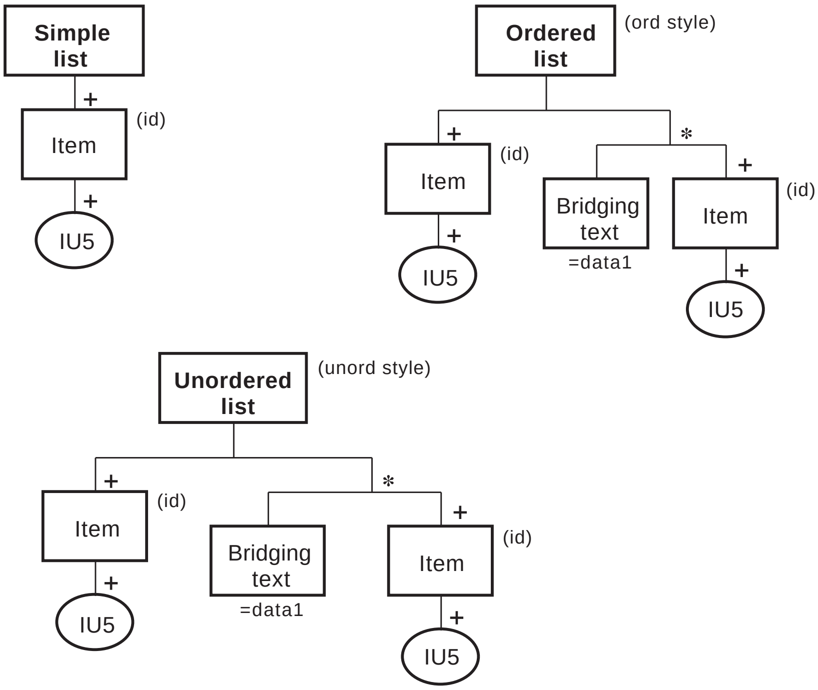


A.3 Tree Diagrams for the Information Pool

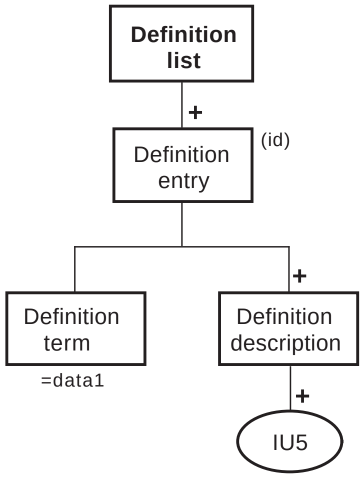
This section collects the tree diagrams for the elements in the information pool.



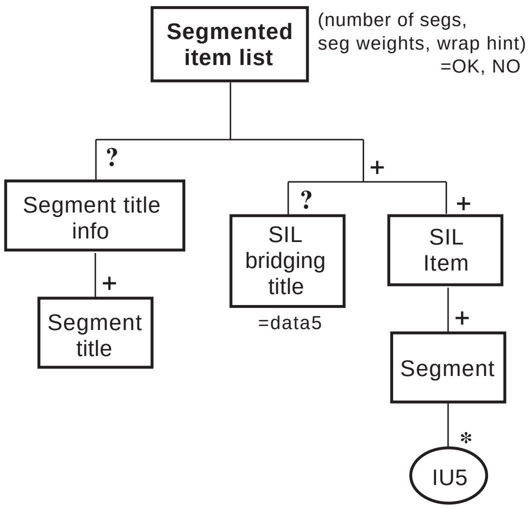
ZK−0663U−R



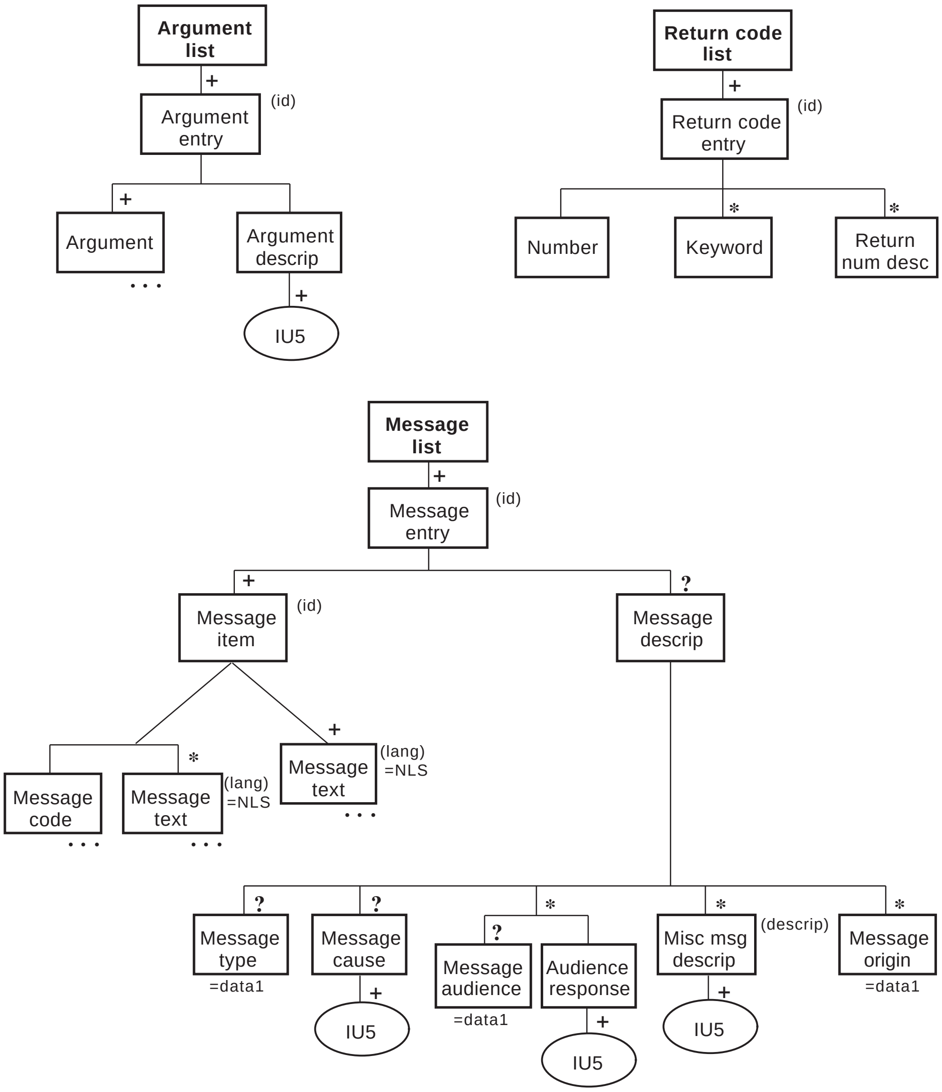
ZK−0653U−R



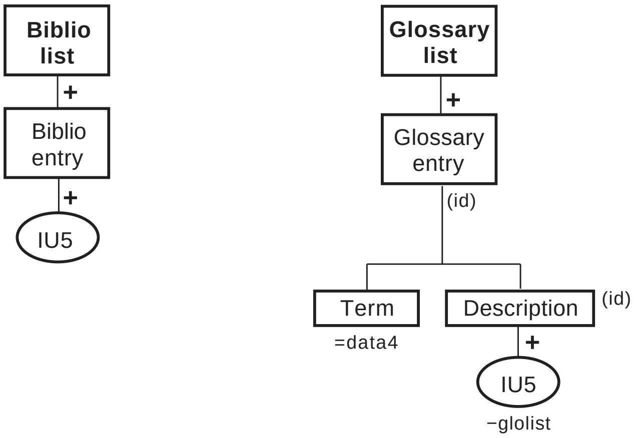
ZK−0669U−R



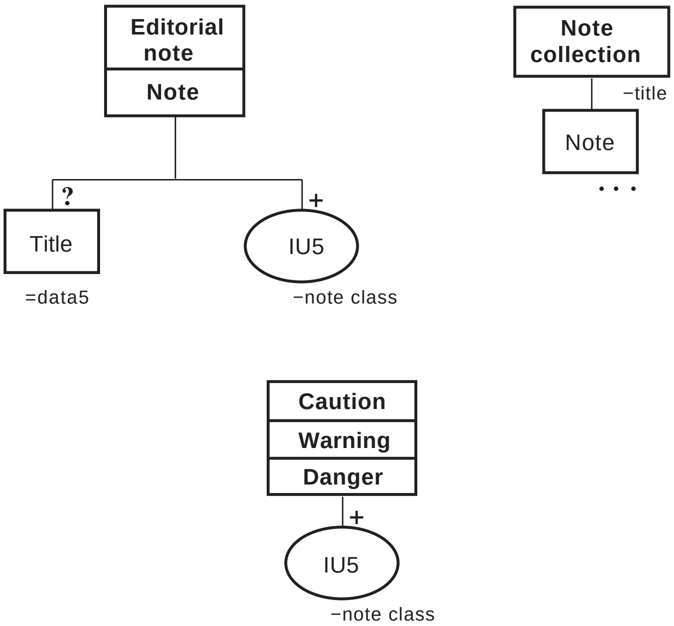
ZK−0654U−R


Argument list
+
Argument entry
(id)
+
Argument
. . .
Argument descrip
+
IU5
Return code list
+
Return code entry
(id)
Number
*
Keyword
*
Return num desc
Message list
+
Message entry
(id)
+
Message item
(id)
?
Message descrip
Message code
. . .
*
Message text
(lang) =NLS
. . .
+
Message text
(lang) =NLS
. . .
?
Message type
=data1
?
Message cause
+
IU5
*
?
Message audience
=data1
Audience response
+
IU5
*
Misc msg descrip
(descrip)
+
IU5
*
Message origin
=data1

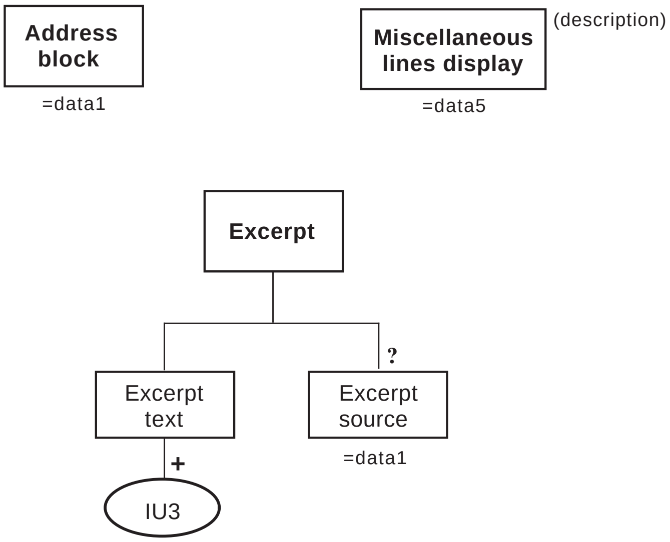
ZK−0652U−R



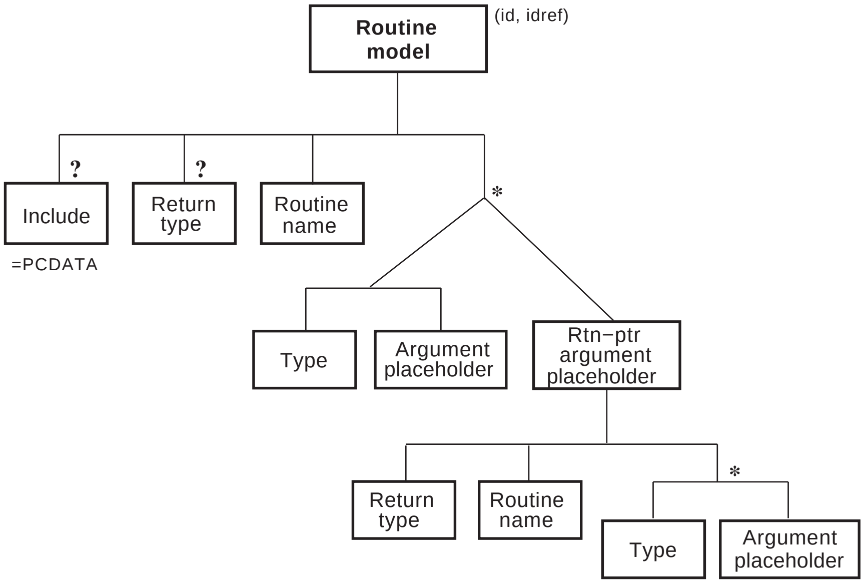
ZK−0676U−R



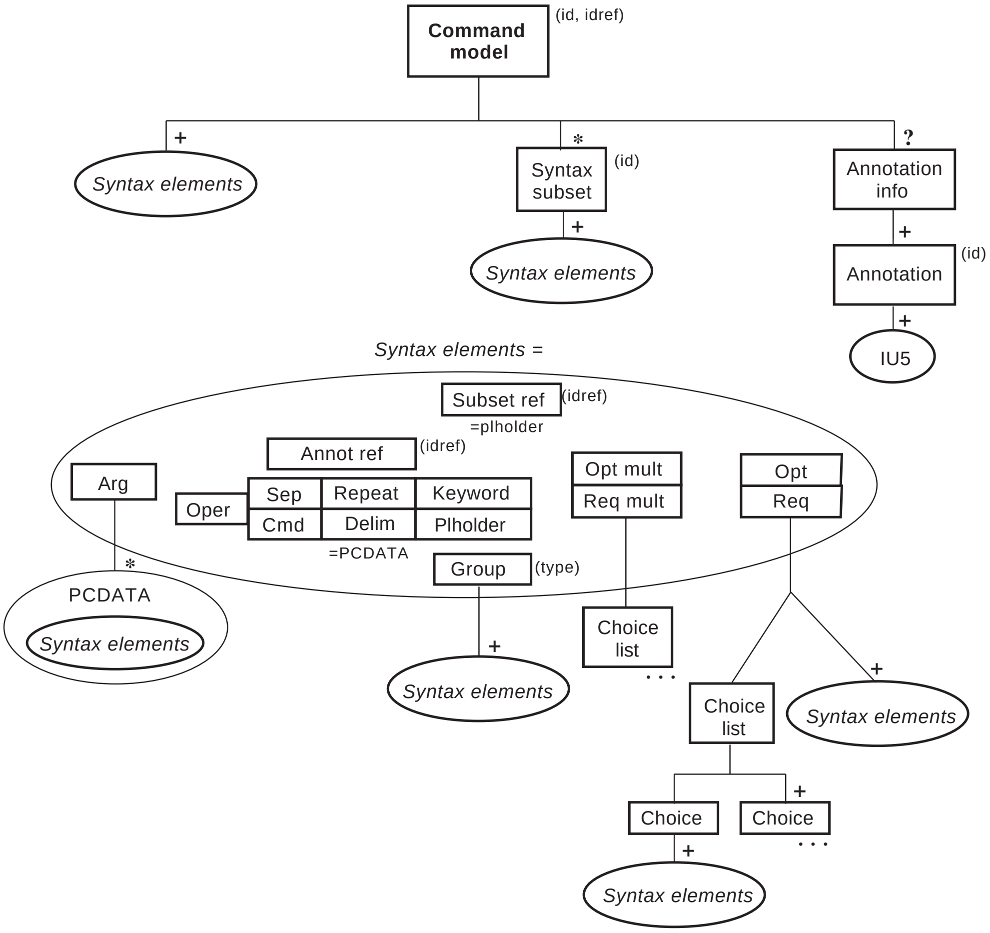
ZK−0662U−R



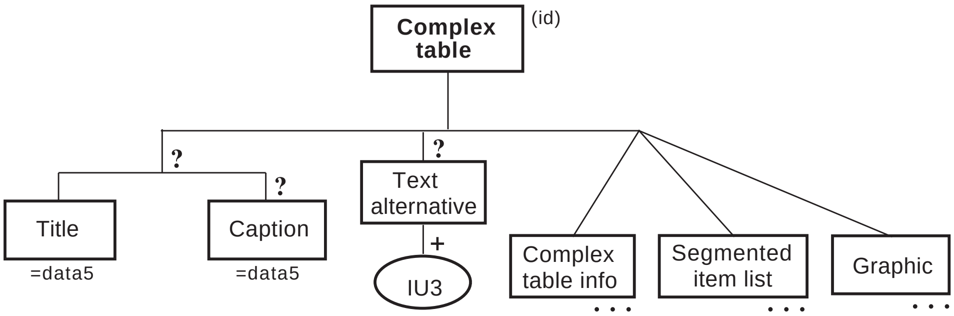
ZK−0668U−R



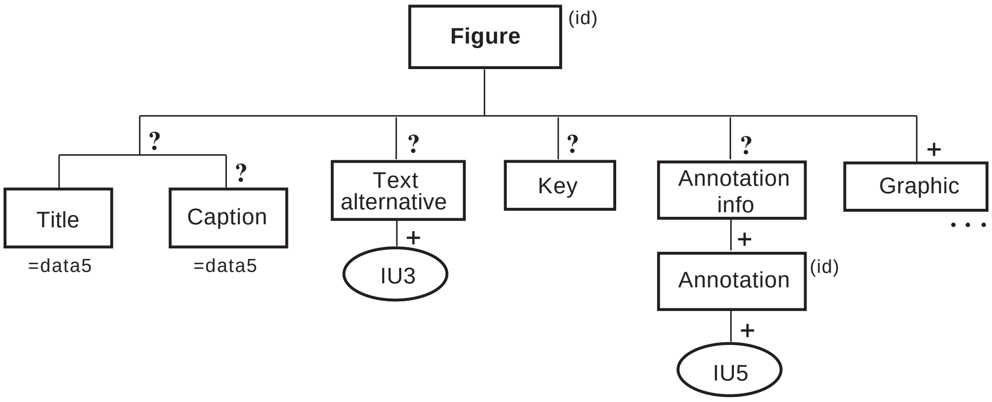
ZK−0665U−R



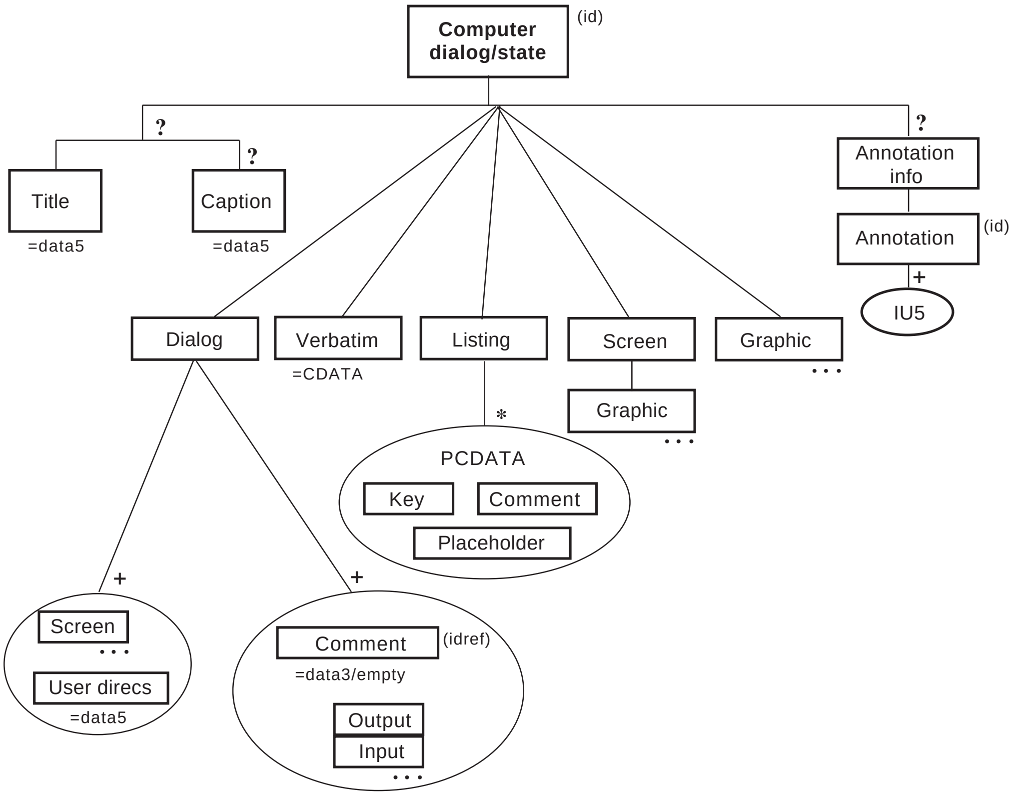
ZK−0666U−R



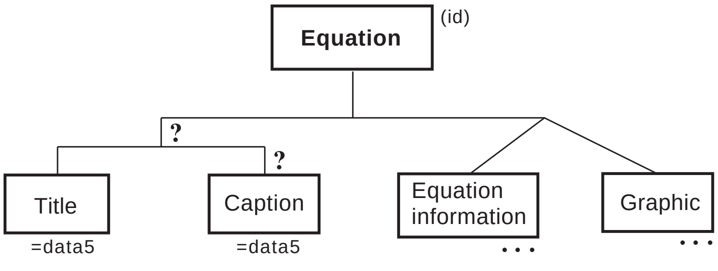
ZK−0659U−R



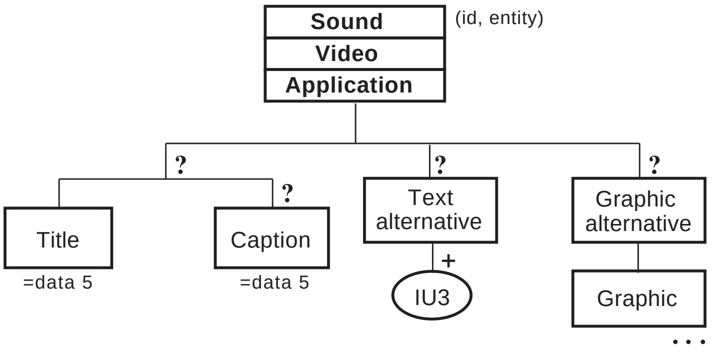
ZK−0658U−R



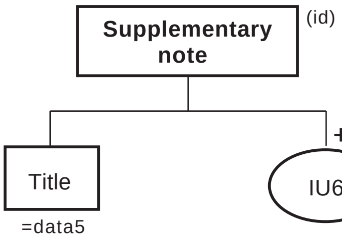
ZK−0656U−R



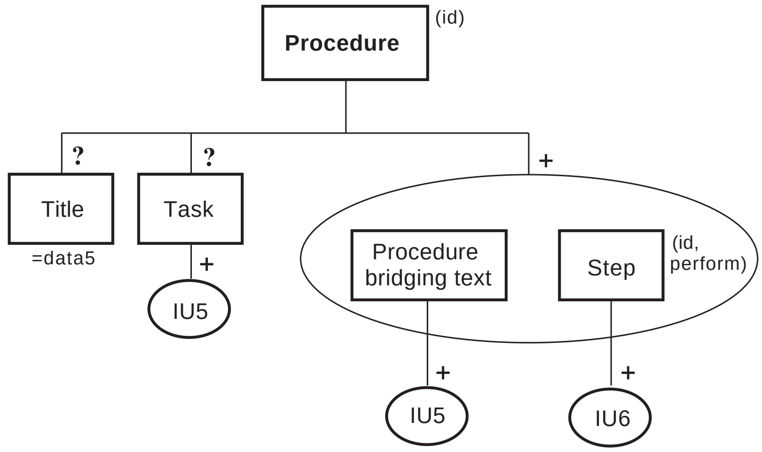
ZK−0660U−R



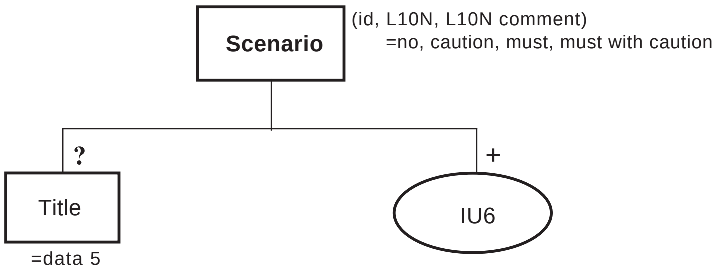
ZK−0661U−R



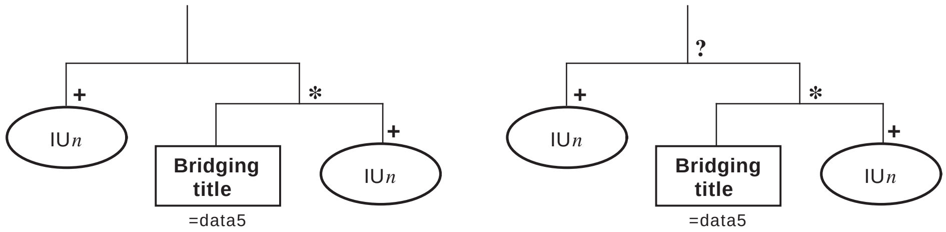
ZK−0664U−R



ZK−0655U−R



ZK−0657U−R



ZK−0667U−R

Handling Document Fragments B

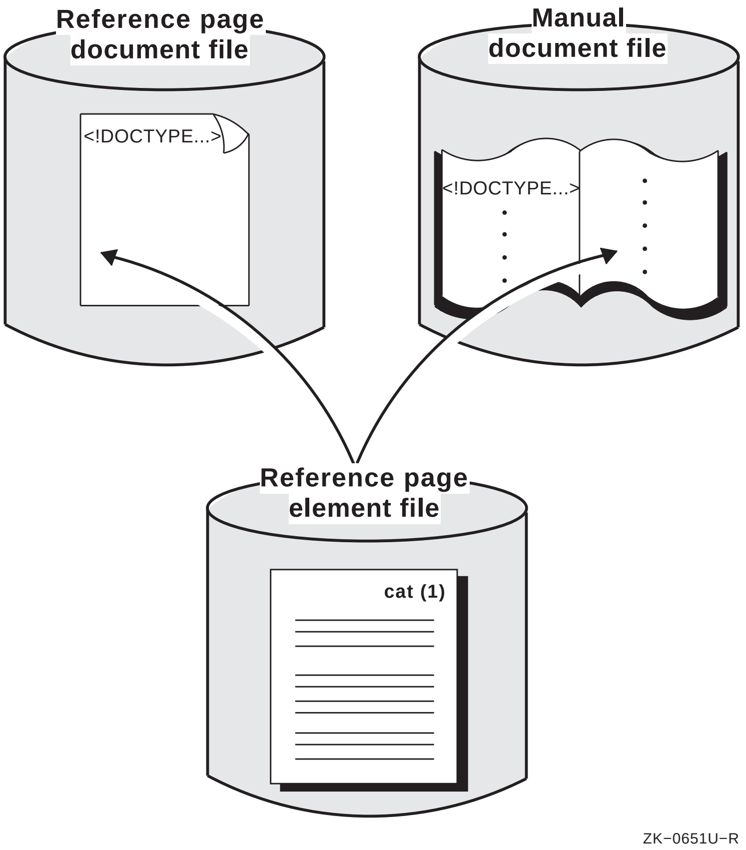
The validation and processing of document fragments presents some problems. There will be times that a document fragment, such as a single chapter, will need to be validated and formatted for output. For example, OSF might require technology suppliers to validate document fragments before sending them to OSF during the development of an offering, or a writer may want to validate and format a single chapter for draft review. However, a single chapter is not a valid document according to the recommendations for the interchange DTD.

A similar problem arises with reference pages. Because of the restriction against using the SUBDOC feature, we have recommended that the reference page element be available in two very different places in a single DTD:

- Directly beneath the top-level document element, for when the page is standalone and serves as the entire document
- In series inside reference divisions, for when a page is part of a collection of reference pages in a manual

Many reference pages are published both as standalone documents (accessible through `man`) and as part of a manual. For every such page, the reference page element (and the element's content) must be stored in a separate file and included, by means of a general entity reference, into two different instance files: a manual-type document instance and a reference page-type document instance. Figure B-1 shows the relationship of the three files.

Figure B-1: Reference Page File Inclusions



All these files are necessary; if the reference page element were stored in the same file with its document type declaration and the top-level document element, the resulting file would contain markup that cannot be included into the context of a reference division deep inside a manual.

It would be convenient to be able to validate and process a document fragment – such as a single chapter, or a file that contains a reference page element – without having to add the required higher-level elements to make the fragment valid.

One way to do this is to create DTD variants that turn various required components, such as metainformation, into optional elements. However, this could result in some information that is essential for processing being optional (for example, if explicit values for certain status attributes in the metainformation are required for proper formatting, but have not been supplied). Also, the number of DTD combinations quickly becomes too extensive to manage realistically.

Ideally the DTD should not be affected by such a fragment-processing feature at all, meaning that an application would somehow have to set a valid context for the fragment instead. This context-setting capability must include a specification of the document type that the fragment ‘‘conforms to,’’ the definition of any necessary entities, and the specification of any elements that are required to put the fragment in a valid context.

One means for accomplishing can be used with processing systems that save a state or an internal representation of a document (which in some systems is a ‘‘tree’’ structure); the system can graft the fragment’s internal representation into the complete document’s representation, allowing the fragment to be validated and processed. (The association between this context-setting processing state and the document fragment is not part of the document; the system provides it.)

Another alternative, rather than using an internal representation of the document, uses an actual file that fills in the missing context. The system matches the fragment with the temporary document file by an entity-like means.

In both solutions, it is important that the fragment and the context-setting component be saved separately from each other, and that they be connected by an application (perhaps controlled by a command-line option), rather than by SGML means such as entities. Otherwise, the fragment must be manually altered when included in its real document context.

These solutions have different strengths and weaknesses. If fragment validation is a necessity for OSF document interchange, the issues surrounding this problem will need to be examined more carefully.

Prototype of the Recommendations C

The design group prototyped its recommendations to test their feasibility. Section C.1 of this appendix contains listings of the resulting DTD code, which was parsed successfully with a small sample document instance on internally developed IBM and Digital validating parsers.

Prototyping was a valuable exercise; it helped to locate inconsistencies, contradictions, and unclear recommendations. The prototype DTD code may also demonstrate to implementors some aspects of the recommendations more clearly than the text and tree diagrams in this document do, and the code could even be used as a starting point for more complete DTD implementations. However, it is incomplete and untested, and is not appropriate for direct use in its current state. Further, it requires some changes to the SGML declaration proposed in the IAWG's SGML Declarations and Parsers position paper, as Section C.2 explains.

To request online copies of any of these files or the sample instance used in validation, contact Eve Maler (Digital Equipment Corporation, Nashua, NH, USA): telephone 603/881–0310, FAX 603/881–0120, or Internet address `maler@zk3.dec.com`.

C.1 Prototype DTD Code

Example C-1 lists the DTD code for the manual document type, Example C-2 lists the code for the reference page document type, and Example C-3 lists the code for the information pool.

The DTD conforms to the design recommendations, embellishing in a few places where appropriate (for example, attribute values and defaults). However, it is not intended to be the ultimate implementation of an OSF interchange DTD. For example, OSF may discover further component requirements during testing of its own implementation. Also, the prototype DTD does assign element and attribute names, as it must, but these names are not intended to be recommendations for real names.

The components needing further study and development are detailed in Section 2.3; the DTD uses parameter entities for several of these components, such as the complex table encoding, so that they can be filled in later. For most others, it implements the recommendations as far as they go and does not make much significant addition to them.

The relationships of the three DTD files are as follows:

- The manual document type makes up the main DTD file; it pulls in the information pool and the reference page module.
- The information pool is stored separately from the document type for reusability purposes; the pool therefore does not depend on any document type that includes it.

- The manual and reference page document types depend on the common attributes and elements defined in the information pool. Therefore, the information pool module is pulled in before the reference page module and before any major declarations in the manual document type.
- The reference page module, being a simulated DTD within the main DTD code, also depends on various attributes and elements defined in the manual module. Therefore, any such attributes are declared before the point at which the reference page module is pulled in.

  If the simulated reference page DTD needs to be independent of the main DTD, it needs to declare new parameter entities with the same functions and it cannot share any elements with the manual document type.
- The manual DTD depends on the declaration of the reference page element provided by the reference page module, but depends on no other characteristic of that module.

Example C-1: Manual Document Type Prototype Code (`compman.dtd`)

```
<!--==================================PROTOTYPE DTD FOR OSF COMPUTER MANUALS-->
<!--       Contact Eve Maler, 603/881-0310, maler@zk3.dec.com, for more info-->
<!--                                         Prototype V1.0, 30 July 1992-->
<!--                                                                     -->
<!--       IMPORTANT: See accompanying recommendation document           -->
<!--       for more information about holes and design choices.          -->
<!--       Search for "TBS" to find holes.                               -->

<!--========================================TO-BE-SPECIFIED (TBS) COMPONENTS-->
<!--                                                                     -->
<!--       These are in addition to those in the included DTD modules.   -->
<!--                                                                     -->
<!--       TOCcon:    Augmentable table of contents information.         -->
<!--       indexcon:  Augmentable index information.                     -->
<!--                                                                     -->
<!ENTITY % TOCcon    "#PCDATA">
<!ENTITY % indexcon  "#PCDATA">

<!--==============================================================NOTATIONS-->

<!--TBS; see recommendation document.-->

<!--======================================================TOP-LEVEL ELEMENT-->

<!ELEMENT CompMan  - O ((MetaInfo, Introductory?, Body, Supplementary?)
                        |RefPage)>

<!--======================================================COMMON ATTRIBUTES-->
<!--                                                                     -->
<!--       These may be used in the reference page "DTD".                -->
<!--       See the information pool module for the definitions of        -->
<!--       additional common attributes that are used in this file.      -->
<!--                                                                     -->
<!--       divident: Implied ID for a division.                          -->
<!--       sequence: POSIX-type conformance sequencing.                  -->
<!--                                                                     -->
<!ENTITY % divident   "id       ID      #IMPLIED">
<!ENTITY % sequence   "seq      NUMBER  #IMPLIED">

<!--==================================================DTD MODULE INCLUSIONS-->
<!--                                                                     -->
<!--       infopool: Pool of info units and data; supplies many          -->
<!--                 parameter entities used in this file.  Must be      -->
<!--                 included before any other DTD modules.              -->
<!--       refpage:  Built-in "document type" for reference pages.       -->
<!--                 Supplies the definitions for RefPage element.       -->
```

Example C-1: (continued)

```
<!--                                                                       -->
<!ENTITY % infopool SYSTEM "compinfo.dtd">
%infopool;

<!ENTITY % refpage SYSTEM "refpage.dtd">
%refpage;

<!--=============================================================COMMON ELEMENTS-->

<!ELEMENT AltTitle - O (%data2;)*>

<!--=============================================================METAINFORMATION-->

<!ELEMENT MetaInfo - O (DocID, ProdID?, Notices, NavInfo?, LinkInfo?,
                        Summary?, Biblio?, Glossary?)>
<!ATTLIST MetaInfo     numvols  NUMBER   #IMPLIED
                       voldivs  IDREFS   #IMPLIED>

    <!--==============================================DOCUMENT ID INFORMATION-->

<!ELEMENT DocID    - O (DocTitle, DocAltTitle?, ManualType?, Library?,
                        OrderNo?, ISBN?, UpdHist?, ValDate?, TranHist?,
                        Abstract?, RetKeys?, AuthInfo?, Reviewers?)>
<!ATTLIST DocID        %divident;
                       status   (DRAFT|SNAP|FINAL) #IMPLIED
                       conform  (USESEQ|NORMAL)    #IMPLIED --e.g.posix conf--
                       version  CDATA              #REQUIRED
                       secure   (MEMONLY|OPEN)     #IMPLIED
                       lang     NAME               #IMPLIED
                       origlang NAME               #IMPLIED
                       origctry NAME               #IMPLIED>
<!ELEMENT DocTitle - O (%data5;)*>
<!ELEMENT DocAltTitle
                   - O (%data2;)*>
<!ELEMENT ManualType
                   - O (#PCDATA)>
<!ELEMENT Library  - O (%data5;)*>
<!ELEMENT OrderNo  - O (#PCDATA)>
<!ELEMENT ISBN     - O (#PCDATA)>
<!ELEMENT UpdHist  - O (%simplemix;)+ -(%complex;)>
<!ELEMENT ValDate  - O (%datecon;)>
<!ATTLIST ValDate      %dateatt;>
<!ELEMENT TranHist - O (%simplemix;)+ -(%complex;)>
<!ELEMENT Abstract - O (%p;)>
<!ELEMENT RetKeys  - O (#PCDATA)>
<!ELEMENT AuthInfo - O (%simplemix-addr;)+ -(%addrcomplex;)>
<!ELEMENT Reviewers
                   - O (%simplemix-addr;)+ -(%addrcomplex;)>

    <!--===============================================PRODUCT ID INFORMATION-->
<!--                                                                       -->
<!--      The following ensures that ProdID is not empty.                  -->
<!--                                                                       -->
<!ELEMENT ProdID   - O ( (Product+, OS*, ReqSW*, ReqHW*)
                        |(OS+, ReqSW*, ReqHW*)
                        |(ReqSW+, ReqHW*)
                        |(ReqHW+))>
<!ATTLIST ProdID       SWlang (%nlslist;)            #IMPLIED>
<!ELEMENT Product  - O (PIDName, PIDVersion)>
<!ATTLIST Product      type (SW|HW|OS|SVC|PROG|MISC) #IMPLIED
                       secure (B1|C2|NONE)           #IMPLIED>
<!ELEMENT OS       - O (PIDName, PIDVersion)>
<!ELEMENT ReqSW    - O (PIDName, PIDVersion)>
<!ELEMENT ReqHW    - O (PIDName, PIDVersion)>
<!ELEMENT PIDName  - O (%data3;)*>
<!ELEMENT PIDVersion
                   - O (%data3;)*>

    <!--==============================================================NOTICES-->

<!ELEMENT Notices  - O (Disclaim?, Safety?, CopyInfo, Security?,
```

Example C-1: (continued)

```
                         TMInfo?, License?, GovRts?, FCCWarn?,
                         Acknowl?, MiscNotice*)>

<!ELEMENT Disclaim - O (%simplecon-title;) -(%complex;)>
<!ELEMENT Safety   - O (%simplecon-title;) -(%complex;)>
<!ELEMENT CopyInfo - O (CopyStmt+)>
<!ELEMENT CopyStmt - O (%data2;)*>
<!ATTLIST CopyStmt      regno    CDATA     #IMPLIED
                        date     CDATA     #REQUIRED
                        owner    CDATA     #REQUIRED>
<!ELEMENT Security - O (%simplecon-title;) -(%complex;)>
<!ELEMENT TMInfo   - O (TMAttrib+)>
<!ELEMENT TMAttrib - O (%data4;)*>
<!ATTLIST TMAttrib      %ident;
                        type     (TM|R|SM) #IMPLIED>
<!ELEMENT License  - O (%simplecon-title;) -(%complex;)>
<!ELEMENT GovRts   - O (%simplecon-title;) -(%complex;)>
<!ELEMENT FCCWarn  - O (%simplecon-title;) -(%complex;)>
<!ELEMENT Acknowl  - O (%simplecon-title;) -(%complex;)>
<!ELEMENT MiscNotice
                   - O ((%title;)?, (%simplemix;)+) -(%complex;)>
<!ATTLIST MiscNotice    %descrip;>

    <!--==================================================OTHER METAINFORMATION-->
<!--                                                                         -->
<!--      The following ensures that NavInfo is not empty.                   -->
<!--                                                                         -->
<!ELEMENT NavInfo  - O ((TOCInfo, IndexInfo?, LinkInfo?)
                        |(IndexInfo, LinkInfo?)
                        |LinkInfo)>
<!ELEMENT TOCInfo  - O (%TOCcon;)>
<!ELEMENT IndexInfo
                   - O (%indexcon;)>
<!--                                                                         -->
<!--      The following ensures that LinkInfo is not empty.                  -->
<!--                                                                         -->
<!ELEMENT LinkInfo - O ((DocDesc+, LDesc*, RPDesc*)|(LDesc+, RPDesc*)|RPDesc+)>
<!ELEMENT DocDesc  - O (%data3;)*>
<!ATTLIST DocDesc       id       ID        #REQUIRED
                        rdoc     NAME      #REQUIRED>
<!ELEMENT LDesc    - O (%data3;)*>
<!ATTLIST LDesc         id       ID        #REQUIRED
                        rdoc     IDREF     #IMPLIED>
<!ELEMENT RPDesc   - O (%data3;)*>
<!ATTLIST RPDesc        id       ID        #REQUIRED
                        rrpage   NAME      #REQUIRED>
<!ELEMENT Summary  - O ((%title;), (%divcon;))>
<!ELEMENT Biblio   - O ((%title;), (%noBdivcon;)?,
                        (((%biblist;), (%divcon;)?) | BibDiv+))>
<!ELEMENT BibDiv   - O ((%title;), (%noBdivcon;)?, (%biblist;), (%divcon;)?)>
<!ELEMENT Glossary - O ((%title;), (%noGdivcon;)?, ((%glolist;) | GlosDiv+))>
<!ELEMENT GlosDiv  - O ((%title;), (%glolist;))>

<!ATTLIST (Biblio|BibDiv|Glossary|GlosDiv)
                        %divident;>

<!--=======================================================INTRODUCTORY MATERIAL-->

<!--                                                                         -->
<!--      The following ensures that Introductory contains at least          -->
<!--      Preface, while allowing an optional Amendments and any             -->
<!--      number of MiscPrefaces in any order.                               -->
<!--                                                                         -->
<!ELEMENT Introductory
                   - O (MiscPreface*,
                       ((Preface, MiscPreface*) & (Amendments, MiscPreface*)?))>

<!--                                                                         -->
<!--      The following ensures that any number of PrefDivs can be           -->
<!--      freely interspersed with the specialized preface divisions,        -->
<!--      while ensuring that the specialized divisions can each occur       -->
```

Example C-1: (continued)

```
<!--      only once.                                                  -->
<!--                                                                  -->
<!ELEMENT Preface  - O ((%title;), AltTitle?, (%divcon;)?,
                        (PrefDiv* & (Applicability, PrefDiv*)? &
                        (Audience, PrefDiv*)? & (Conventions, PrefDiv*)? &
                        (DocStructure, PrefDiv*)? & (Feedback, PrefDiv*)? &
                        (Map, PrefDiv*)? & (DocPurpose, PrefDiv*)?))>
<!ATTLIST Preface      %divident;
                       %L10Natt;>
<!ELEMENT PrefDiv  - O ((%title;), AltTitle?, (%divcon;))>
<!ATTLIST PrefDiv      %divident;
                       %descrip;
                       %L10Natt;>
<!ELEMENT Applicability
                   - O ((%title;), AltTitle?, (%divcon;))>
<!ATTLIST Applicability %divident;
                       %L10Natt;>
<!ELEMENT Audience - O ((%title;), AltTitle?, (%divcon;))>
<!ATTLIST Audience     %divident;
                       %L10Natt;>
<!ELEMENT Conventions
                   - O ((%title;), AltTitle?, (%divcon;))>
<!ATTLIST Conventions  %divident;
                       %L10Natt;>
<!ELEMENT DocStructure
                   - O ((%title;), AltTitle?, (%divcon;)?, DocMedia*)>
<!ATTLIST DocStructure %divident;
                       %L10Natt;>
<!ELEMENT DocMedia - O ((%title;)?, AltTitle?, (%divcon;))>
<!ATTLIST DocMedia     media    CDATA    #REQUIRED>
<!ELEMENT Feedback - O ((%title;), AltTitle?, (%divcon;))>
<!ATTLIST Feedback     %divident;
                       %L10Natt;>
<!ELEMENT Map      - O ((%title;), AltTitle?, (%divcon;))>
<!ATTLIST Map          %divident;
                       %L10Natt;>
<!ELEMENT DocPurpose
                   - O ((%title;), AltTitle?, (%divcon;))>
<!ATTLIST DocPurpose   %divident;
                       %L10Natt;>
<!ELEMENT MiscPreface
                   - O ((%title;), AltTitle?, (%divcon;)?, PrefDiv*)>
<!ATTLIST MiscPreface  %divident;
                       %descrip;
                       %L10Natt;>
<!ELEMENT Amendments
                   - O ((%title;), AltTitle?, (%divcon;)?, PrefDiv*)>
<!ATTLIST Amendments   %divident;
                       %L10Natt;>

<!--==========================================================DOCUMENT BODY-->

<!ELEMENT Body     - O ((SuperPart, SuperPart+) | ((IntroDiv|Part), Part+)
                        |(Div1|Reference|Messages)+)>
<!ELEMENT SuperPart
                   - O ((%title;), AltTitle?, Part+)>
<!ELEMENT Part     - O (NavInfo?, Summary?, Biblio?, (%title;), AltTitle?,
                        (%divcon;)?, (Div1|Reference|Messages)+)>
<!ELEMENT IntroDiv - O (TOCInfo?, Summary?, Biblio?, (%title;), AltTitle?,
                        (%divcon;)?, (Div2|Reference|Messages)*)>
<!ELEMENT Div1     - O (TOCInfo?, Summary?, Biblio?, (%title;), AltTitle?,
                        (%divcon;)?, (Div2|Reference|Messages)*)>
<!ELEMENT Div2     - O ((%title;), AltTitle?,
                        (%divcon;)?, (Div3|Reference|Messages)*)>
<!ELEMENT Div3     - O ((%title;), AltTitle?,
                        (%divcon;)?, (Div4|Reference|Messages)*)>
<!ELEMENT Div4     - O ((%title;), AltTitle?,
                        (%divcon;)?, (Div5|Reference|Messages)*)>
<!ELEMENT Div5     - O ((%title;), AltTitle?, (%divcon;)?, Div6*)>
<!ELEMENT Div6     - O ((%title;), AltTitle?, (%divcon;)?, Div7*)>
<!ELEMENT Div7     - O ((%title;), AltTitle?, (%divcon;))>
```

Example C-1: (continued)

```
<!ELEMENT Messages - O ((%title;), AltTitle?, (%noMdivcon;)?,
                        (((%msglist;), (%divcon;)?) | MsgDiv1+))>
<!ELEMENT MsgDiv1  - O ((%title;), AltTitle?, (%noMdivcon;)?,
                        (((%msglist;), (%divcon;)?) | MsgDiv2+))>
<!ELEMENT MsgDiv2  - O ((%title;), AltTitle?, (%noMdivcon;)?,
                        ((%msglist;), (%divcon;)?))>

<!ELEMENT Reference
                   - O ((%title;), AltTitle?, (%divcon;)?,
                        (RefDiv1+|RefPage+))>
<!ELEMENT RefDiv1  - O ((%title;), AltTitle?, (%divcon;)?,
                        (RefDiv2+|RefPage+))>
<!ELEMENT RefDiv2  - O ((%title;), AltTitle?, (%divcon;)?, RefPage+)>

<!ATTLIST (SuperPart|Part|IntroDiv|Div1|Div2|Div3|Div4|Div5|Div6|Div7
           |Reference|RefDiv1|RefDiv2|Messages|MsgDiv1|MsgDiv2)
                       %divident;
                       %L10Natt;
                       %sequence;>

<!--==================================================SUPPLEMENTARY MATERIAL-->

<!ELEMENT Supplementary
                   - O (Part | (Div1|Reference|Messages)+)>
```

Example C-2: Reference Page Document Type Prototype Code (`refpage.dtd`)

```
<!--==================================PROTOTYPE DTD FOR OSF REFERENCE PAGES-->
<!--       Contact Eve Maler, 603/881-0310, maler@zk3.dec.com, for more info-->
<!--                                        Prototype V1.0, 30 July 1992-->
<!--                                                                    -->
<!--       IMPORTANT: See accompanying recommendation document          -->
<!--       for more information about holes and design choices.         -->
<!--       Search for "TBS" to find holes.                              -->

<!ELEMENT RefPage  - O (Brief, RPBody, Refs?, About?)>
<!ATTLIST RefPage      %divident;
                       category (CMD|ADMCMD|CALL|RTN|FILE|DEVICE|STMT
                                 |TAG|MISC)          #REQUIRED
                       audience (USER|ADMIN|PROG)
                                                     #REQUIRED
                       date     CDATA                #IMPLIED
                       version  CDATA                #IMPLIED
                       contents (BASIC|ENHANCED)     #IMPLIED>

<!ELEMENT Brief    - O (NameInfo, Purpose, Permuted?, Model, Scope?)>
<!ELEMENT NameInfo - O (FamName?, (Name, Alias*, Purpose?)+)>
<!ELEMENT FamName  - O (%data5;)*>
<!ELEMENT Name     - O (%data5;)*>
<!ATTLIST Name         %ident;>
<!ELEMENT Alias    - O (%data5;)*>
<!ELEMENT Purpose  - O (%data5;)*>
<!ELEMENT Permuted - O (#PCDATA)>
<!ELEMENT Model    - O (%models;)+>
<!ELEMENT Scope    - O ((%title;), (%divcon;))>

<!ELEMENT RPBody   - O (Description, Params?, Options?, Keys?, Fields?,
                        Subcommands?, Types?, Functions?, Results?,
                        RetCodes?, Macros?, Constants?, Files?, Restrictions?,
                        Limitations?, Examples?, Subject*)>

<!ELEMENT Refs     - O (SeeAlso)>
<!ELEMENT SeeAlso  - O (%divcon;)>

<!ELEMENT About    - O (SWAcknowls?, SWSecurity?,
                        SWLicense?, SWAuthors?, SWHistory?,
```

Example C-2: (continued)

```
                        Copyrights?, Trademarks?, Subject*)>

<!ELEMENT Subject  - O ((%title;), (%divcon;))>
<!ATTLIST Subject      %descrip;
                       rname    IDREF    #IMPLIED>

<!ELEMENT (Description|Params|Options|Keys|Fields|Subcommands|Types
           |Functions|Results|RetCodes|Macros|Constants|Files|Restrictions
           |Limitations|Examples|SWAcknowls|SWSecurity|SWLicense|SWAuthors
           |SWHistory|Copyrights|Trademarks)
                   - O (%divcon;)>
<!ATTLIST (Description|Params|Options|Keys|Fields|Subcommands|Types
           |Functions|Results|RetCodes|Macros|Constants|Files|Restrictions
           |Limitations|Examples|SWAcknowls|SWSecurity|SWLicense|SWAuthors
           |SWHistory|Copyrights|Trademarks)
                       rname    IDREF    #IMPLIED>
```

Example C-3: Information Pool Prototype Code (`compinfo.dtd`)

```
<!--=================================PROTOTYPE OSF COMPUTER INFORMATION POOL-->
<!--       Contact Eve Maler, 603/881-0310, maler@zk3.dec.com, for more info-->
<!--                                          Prototype V1.0, 30 July 1992-->
<!--                                                                      -->
<!--       IMPORTANT: See accompanying recommendation document            -->
<!--       for more information about holes and design choices.          -->
<!--       Search for "TBS" to find holes.                                -->

<!--========================================TO-BE-SPECIFIED (TBS) COMPONENTS-->
<!--                                                                      -->
<!--       nlslist: National Language Support keywords.                   -->
<!--       tabcon:  Complex table encoding.                               -->
<!--       mathcon: Equation encoding.                                    -->
<!--                                                                      -->
<!ENTITY % nlslist        "TBS">
<!ENTITY % tabcon         "TBS">
<!ENTITY % mathcon        "TBS">
<!ENTITY % idxentrycon    "TBS">
<!ENTITY % structmodelcon "TBS">
<!ELEMENT TBS         - - (#PCDATA)>

<!--=====================================================COMMON CONTENT MODELS-->

<!ENTITY % datecon "#PCDATA">

<!--       NAME INDIRECTION FOR HIGH-LEVEL DTDS:                          -->
<!--                                                                      -->
<!ENTITY % p       "P">
<!ENTITY % biblist "BibList">
<!ENTITY % glolist "GlosList">
<!ENTITY % msglist "MsgList">
<!ENTITY % title   "Title">

<!--       IU CLASSES AND MIXTURES:                                       -->
<!--                                                                      -->
<!--       "Mixture" entities are free mixtures of elements that can      -->
<!--       take the usual + or * occurrence indicators where they appear.  -->
<!--       "Content" entities are mixtures that have some internal        -->
<!--       organization because of the presence of bridging titles.       -->
<!--       They take either no occurrence indicator (because something    -->
<!--       inside it is required), or an indicator of ? (because zero     -->
<!--       elements in its content are required).                         -->
<!--                                                                      -->
<!--       IU classes:                                                    -->
<!--                                                                      -->
<!--                 Generalized lists                                    -->
<!--                 Specialized definition lists                         -->
<!--                 Notes                                                -->
<!--                 Text blocks                                          -->
```

Example C-3: (continued)

```
<!--                Computer language models                               -->
<!--                Static illustrations                                   -->
<!--                Dynamic illustrations                                  -->
<!--                All illustrations                                      -->
<!--                Division-like elements                                 -->
<!--                                                                       -->
<!ENTITY % genlists  "SList|OList|UList|DefList|SegList">
<!ENTITY % speclists "ArgList|RetList|MsgList|BibList|GlosList">
<!ENTITY % notes     "Note|Caution|Warning|Danger|EdNote|Notes">
<!ENTITY % blocks    "AddrBlock|Display|Excerpt">
<!ENTITY % models    "RtnModel|CmdModel|FileModel|StructModel">
<!ENTITY % statills  "Table|Figure|Computer|Equation">
<!ENTITY % dynills   "Sound|Video|Applic">
<!ENTITY % ills      "%statills;|%dynills;">
<!ENTITY % divlikes  "SuppNote|Procedure|Scenario">
<!--                                                                       -->
<!--       Convenience entities for content:                               -->
<!--                                                                       -->
<!--       divmix:     IU mixture for divisions, except no br titles.      -->
<!--       divlikemix: IU mixture for div-likes, except no br titles.      -->
<!--       noBdivmix:  Normal division mix, except no bib list.            -->
<!--       noGdivmix:  Normal division mix, except no glos list.           -->
<!--       noMdivmix:  Normal division mix, except no message list.        -->
<!--       complex:    IUs disallowed in simple mixes, for exclusions.     -->
<!--       addrcomplex:IUs disallowed in simple-addr mixes, for exclusions. -->
<!--                                                                       -->
<!ENTITY % divmix      "P|%genlists;|%speclists;|%notes;|%blocks;|%models;
                        |%ills;|%divlikes;">

<!ENTITY % noBdivmix   "P|%genlists;|ArgList|RetList|MsgList|GlosList
                        |%notes;|%blocks;|%models;|%ills;|%divlikes;">

<!ENTITY % noGdivmix   "P|%genlists;|ArgList|RetList|MsgList|BibList
                        |%notes;|%blocks;|%models;|%ills;|%divlikes;">

<!ENTITY % noMdivmix   "P|%genlists;|ArgList|RetList|BibList|GlosList
                        |%notes;|%blocks;|%models;|%ills;|%divlikes;">

<!ENTITY % complex     "%speclists;|%notes;|%blocks;|%models;|%ills;">

<!ENTITY % addrcomplex "%speclists;|%notes;|Display|Excerpt|%models;|%ills;">
<!--                                                                       -->
<!--       IU entities actually used in content models:                    -->
<!--                                                                       -->
<!--       simplemix:       IU2                                            -->
<!--       simplemix-addr:  IU4                                            -->
<!--       simplecon-title: IU3 (IU2 with bridging titles)                 -->
<!--       IUmix:           IU5                                            -->
<!--       paramix:         IU5 minus P, notes, dynills, speclists         -->
<!--       divlikecon:      IU6 (IU5 with bridging titles)                 -->
<!--       divcon:          IU1 (IU6 with bridging titles)                 -->
<!--       noBdivcon:       IU6, except no bibliography list.              -->
<!--       noGdivcon:       IU6, except no glossary list.                  -->
<!--       noMdivcon:       IU6, except no message list.                   -->
<!--                                                                       -->
<!ENTITY % simplemix       "P|%genlists;">
<!ENTITY % simplemix-addr  "P|%genlists;|AddrBlock">
<!ENTITY % simplecon-title "(%simplemix;)+, (BrTitle, (%simplemix;)+)*">

<!ENTITY % IUmix           "P|%genlists;|%speclists;|%notes;|%blocks;
                            |%models;|%ills;">

<!ENTITY % paramix         "%genlists;|%blocks;|%models;|%statills;">
<!ENTITY % divlikecon      "(%IUmix;)+, (BrTitle, (%IUmix;)+)*">
<!ENTITY % divcon          "(%divmix;)+, (BrTitle, (%divmix;)+)*">
<!ENTITY % noBdivcon       "(%noBdivmix;)+, (BrTitle, (%noBdivmix;)+)*">
<!ENTITY % noGdivcon       "(%noGdivmix;)+, (BrTitle, (%noGdivmix;)+)*">
<!ENTITY % noMdivcon       "(%noMdivmix;)+, (BrTitle, (%noMdivmix;)+)*">

<!--       DATA CLASSES AND MIXTURES:                                      -->
<!--                                                                       -->
```

Example C-3: (continued)

```
<!--       Data classes:                                                   -->
<!--                                                                       -->
<!--                 Equation data                                         -->
<!--                 Graphic data                                          -->
<!--                 General key data                                      -->
<!--                 Technical key data                                    -->
<!--                 Cross-reference links from data                       -->
<!--                 Footnote data (no entity)                             -->
<!--                 General links from data                               -->
<!--                                                                       -->
<!ENTITY % eqndata   "EqnData">
<!ENTITY % graphdata "Graphic">

<!ENTITY % genkeys   "Acro|DocCit|Author|Term|NewTerm|Emph|Phrase|Quote|Sub
                      |Super|TM|Address|Date|Time|Calendar|Phone|Honorific
                      |Unit|PrName|MiscData">

<!ENTITY % techkeys  "String|Const|Number|Not|Key|Win|Arg|Kwd|Cmd|Input
                      |Output|Var|MsgCode|Msg|Rtn">

<!ENTITY % xrefs     "XRef|DocRef|RPageRef">
<!ENTITY % dlinks    "Spot|IdxSpot|Link">
<!--                                                                       -->
<!--       Data mixtures:                                                  -->
<!--                                                                       -->
<!ENTITY % data1 "#PCDATA|%eqndata;|%graphdata;|%genkeys;|%techkeys;
                  |%xrefs;|Footnote|%dlinks;">

<!ENTITY % data2 "#PCDATA|%genkeys;">
<!ENTITY % data3 "#PCDATA|%genkeys;|%techkeys;">
<!ENTITY % data4 "#PCDATA|%graphdata;|%genkeys;|%techkeys;|Footnote">
<!ENTITY % data5 "#PCDATA|%eqndata;|%genkeys;|%techkeys;|%dlinks;">

<!--       SYNTAX ELEMENT MIXTURE:                                         -->
<!--                                                                       -->
<!--     % synels                                     (see command models)-->

<!--==========================================COMMON ATTRIBUTES AND ELEMENTS-->
<!--                                                                       -->
<!--       These attributes are also used in high-level DTDs that          -->
<!--       include the information pool module.                            -->
<!--                                                                       -->
<!--       ident:   Implied ID attribute for info-level elements.          -->
<!--       L10Natt: Two localization attributes; used on                   -->
<!--                info-level and div-level elements.                     -->
<!--       descrip: Required attribute for miscellaneous elements.         -->
<!--       dateatt: Date attributes; used in date-like elements.           -->
<!--                                                                       -->
<!ENTITY % ident      "id      ID                #IMPLIED">

<!ENTITY % L10Natt    "L10N    (NO|CAUTION|MUST) #IMPLIED
                       L10Ncom CDATA             #IMPLIED">

<!ENTITY % descrip    "desc    CDATA             #REQUIRED">

<!ENTITY % dateatt    "month   NUMBER            #REQUIRED
                       dayofmo NUMBER            #REQUIRED
                       year    NUMBER            #REQUIRED">
<!--                                                                       -->
<!--      Some of these elements are also used in high-level DTDs          -->
<!--      that include the information pool module.                        -->
<!--                                                                       -->
<!ELEMENT Title    - O (%data5;)+>
<!ELEMENT Caption  - O (%data5;)+>
<!ELEMENT TextAlt  - O (%simplemix;)+ -(%complex;)>
<!ELEMENT Legend   - O (%simplemix;)+ -(%complex;)>
<!ELEMENT Annots   - O (Annot+)>
<!ELEMENT Annot    - O (%IUmix;)+>

<!--=====================================================================PARAGRAPH-->
```

Example C-3: (continued)

```
<!ELEMENT P        O O (((%data1;)|(%paramix;))*)>
<!ATTLIST P            %L10Natt;>

<!--======================================================GENERALIZED LISTS-->

<!ELEMENT SList    - - (Item+)>
<!ELEMENT UList    - - (Item+, (ListText, Item+)*)>
<!ATTLIST UList        ustyle   ENTITY   #IMPLIED>
<!ELEMENT OList    - - (Item+, (ListText, Item+)*)>
<!ATTLIST OList        ostyle   (ARABIC|ROMANL|ROMANU|ALPHAL|ALPHAU)
                                          #IMPLIED>
<!ELEMENT Item     - O (%IUmix;)+>
<!ATTLIST Item         %ident;>
<!ELEMENT ListText - O (%data1;)*>

<!ELEMENT DefList  - - (DefEntry)+>
<!ELEMENT DefEntry - O (DefTerm, DefDesc)>
<!ATTLIST DefEntry     %ident;>
<!ELEMENT DefTerm  - O (%data1;)*>
<!ELEMENT DefDesc  - O (%IUmix;)+>

<!ELEMENT SegList  - - (SegTitles?, ((SegBrTitle?, SegItem+)+))>
<!ATTLIST SegList      numsegs  NUMBER           #REQUIRED
                       segwts   NUMBERS          #REQUIRED
                       wrap     (OK|NO)          "OK">
<!ELEMENT SegTitles
                   - O (SegTitle+)>
<!ELEMENT SegTitle - O (%data5;)*>
<!ELEMENT SegBrTitle
                   - O (%data5;)*>
<!ELEMENT SegItem  - O (Seg+)>
<!ELEMENT Seg      - O (%IUmix;)*>

<!--===========================================SPECIALIZED DEFINITION LISTS-->

<!ELEMENT ArgList  - - (ArgEntry+)>
<!ELEMENT ArgEntry - O (Arg+, ArgDesc)>
<!ATTLIST ArgEntry     %ident;>
<!--      Arg                                        (see tech key data)-->
<!ELEMENT ArgDesc  - O (%IUmix;)+>

<!ELEMENT RetList  - - (RetEntry+)>
<!ELEMENT RetEntry - O (RetValue, RetKwd*, RetDesc*)>
<!ATTLIST RetEntry     %ident;>
<!ELEMENT RetValue - O (#PCDATA)>
<!ELEMENT RetKwd   - O (#PCDATA)>
<!ELEMENT RetDesc  - O (%IUmix;)+>

<!ELEMENT MsgList  - - (MsgEntry+)>
<!ELEMENT MsgEntry - O (MsgItem+, MsgInfo?)>
<!ATTLIST MsgEntry     %ident;>
<!ELEMENT MsgItem  - O ((MsgCode, Msg*) | Msg+)>
<!ATTLIST MsgItem      %ident;>
<!--      MsgCode                                    (see tech key data)-->
<!--      Msg                                        (see tech key data)-->

<!--                                                                    -->
<!--      The following ensures that MsgInfo is not empty.              -->
<!--                                                                    -->
<!ELEMENT MsgInfo  - O
                ((MsgType, MsgCause?, (MsgAud?, MsgResp)*, MsgDesc*, MsgOrig*)
                 |(MsgCause, (MsgAud?, MsgResp)*, MsgDesc*, MsgOrig*)
                 |((MsgAud?, MsgResp)+, MsgDesc*, MsgOrig*)
                 |(MsgDesc+, MsgOrig*)
                 |(MsgOrig+))>
<!ELEMENT MsgType  - O (%data1;)+>
<!ELEMENT MsgCause - O (%IUmix;)+>
<!ELEMENT MsgAud   - O (%data1;)*>
<!ELEMENT MsgResp  - O (%IUmix;)+>
<!ELEMENT MsgDesc  - O (%IUmix;)+>
<!ATTLIST MsgDesc      %descrip;>
```

Example C-3: (continued)

```
<!ELEMENT MsgOrig  - O (%data1;)*>

<!ELEMENT BibList  - - (BibEntry+)>
<!ELEMENT BibEntry - O (%IUmix;)+>

<!ELEMENT GlosList - - (GlosEntry+)>
<!ELEMENT GlosEntry
                   - O (GTerm, GDesc+)>
<!ATTLIST GlosEntry    %ident;>
<!ELEMENT GTerm    - O (%data1;)*>
<!ELEMENT GDesc    - O (%IUmix;)+ -(GlosList)>

<!--===================================================================NOTES-->

<!ELEMENT Note     - - (Title?, (%IUmix;)+) -(%notes;)>
<!ELEMENT EdNote   - - (Title?, (%IUmix;)+) -(%notes;)>
<!--      Title                                       (see common elems)-->
<!ELEMENT Notes    - - (Note+)>

<!ELEMENT Caution  - - ((%IUmix;)+) -(%notes;)>
<!ELEMENT Warning  - - ((%IUmix;)+) -(%notes;)>
<!ELEMENT Danger   - - ((%IUmix;)+) -(%notes;)>

<!--===========================================================ADDRESS BLOCK-->

<!ELEMENT AddrBlock
                   - - (%data1;)+ -(Address)>

<!--=============================================MISCELLANEOUS LINES DISPLAY-->

<!ELEMENT Display  - - (%data5;)*>

<!--=================================================================EXCERPT-->

<!ELEMENT Excerpt  - - (ExcerptText, Source?)>
<!ELEMENT ExcerptText
                   - O (%simplecon-title;) -(%complex;)>
<!ELEMENT Source   - O (%data1;)*>

<!--==================================================================MODELS-->

<!ELEMENT RtnModel - - (Incl*, Type, Rtn, ((Type, Arg) | RtnPtr)*)>
<!ATTLIST RtnModel     %ident;>
<!ELEMENT Incl     - O (#PCDATA)                          --include instrucs-->
<!ELEMENT Type     - O (#PCDATA)                                 --data type-->
<!--      Rtn                                        (see tech key data)-->
<!--      Arg                                        (see tech key data)-->
<!ELEMENT RtnPtr   - - (Type, Rtn, (Type, Arg)*)  --arg that pts to diff rtn-->

<!-- Local entity definition for command models:                           -->
<!--                                                                       -->
<!ENTITY  % synels     "Opt|Req|OptMult|ReqMult|Grp|Rep|Arg|Cmd|Kwd|Var
                        |Del|Op|Sep|SubsetRef|AnnotRef">

<!ELEMENT CmdModel - - ((%synels;)+, Subset*, Annots?)>
<!ATTLIST CmdModel     %ident;>

<!ELEMENT Opt      - - ((%synels;)+|(ChList))>
<!ELEMENT Req      - - ((%synels;)+|(ChList))>
<!ELEMENT OptMult  - - (ChList)>
<!ELEMENT ReqMult  - - (ChList)>
<!ELEMENT ChList   - - (Chc, Chc+)>
<!ELEMENT Chc      - - ((%synels;)+)>
<!ATTLIST Chc          default  (DEF|NONDEF)       "NONDEF" --default status-->
<!ELEMENT Grp      - - ((%synels;)+)>
<!ATTLIST Grp          type     (NORMAL|COMPOSITE) "COMPOSITE">
<!ELEMENT Rep      - - (#PCDATA)>
<!--      Arg                                        (see tech key data)-->
<!--      Cmd                                        (see tech key data)-->
<!--      Kwd                                        (see tech key data)-->
<!--      Var                                        (see tech key data)-->
```

Example C-3: (continued)

```
<!ELEMENT Del      - - (#PCDATA)                                  --delimiter-->
<!ELEMENT Op       - - (#PCDATA)                                   --operator-->
<!ELEMENT Sep      - - (#PCDATA)                                  --separator-->
<!ELEMENT SubsetRef
                   - - (#PCDATA)>
<!ATTLIST SubsetRef    rid      IDREF    #REQUIRED>
<!ELEMENT Subset   - - (%synels;)+>
<!ATTLIST Subset       %ident;>
<!ELEMENT AnnotRef - O EMPTY>
<!ATTLIST AnnotRef     rid      IDREF    #REQUIRED>

<!ELEMENT FileModel
                   - - ((%synels;)+, Subset*, Annots?)>

<!ELEMENT StructModel
                   - - (%structmodelcon;)>

<!--======================================================================TABLE-->

<!ELEMENT Table    - - ((Title, Caption?)?, TextAlt?, Legend?,
                        (%tabcon;|SegList|Graphic))>
<!--      Title                                           (see common elems)-->
<!--      Caption                                         (see common elems)-->
<!--      TextAlt                                         (see common elems)-->
<!--      Legend                                          (see common elems)-->
<!--      SegList                                            (see gen lists)-->
<!--      Graphic                                         (see graphic data)-->

<!--=====================================================================FIGURE-->

<!ELEMENT Figure   - - ((Title, Caption?)?, TextAlt?, Legend?, Annots?,
                        Graphic+)>
<!ATTLIST Figure       %ident;>
<!--      Title                                           (see common elems)-->
<!--      Caption                                         (see common elems)-->
<!--      TextAlt                                         (see common elems)-->
<!--      Legend                                          (see common elems)-->
<!--      Annots                                          (see common elems)-->
<!--      Graphic                                         (see graphic data)-->

<!--=====================================================COMPUTER DIALOG/STATE-->

<!ELEMENT Computer - - ((Title, Caption?)?,
                        (Dialog|Verbatim|Listing|Screen|Graphic), Annots?)>
<!ATTLIST Computer     %ident;>
<!ELEMENT Dialog   - O ((Output|Input|Comment)+ | (Direcs|Screen)+)>
<!--      Title                                           (see common elems)-->
<!--      Caption                                         (see common elems)-->
<!--      Output                                         (see tech key data)-->
<!--      Input                                          (see tech key data)-->
<!ELEMENT Comment  - O (%data3;)*>
<!ATTLIST Comment      rid      IDREF    #CONREF            --points to Annot-->
<!ELEMENT Direcs   - O (%data5;)*>
<!ELEMENT Screen   - O (Graphic)>
<!ELEMENT Verbatim - - CDATA>
<!ELEMENT Listing  - O (#PCDATA|Key|Comment|Var)*>
<!--      Graphic                                         (see graphic data)-->

<!--===================================================================EQUATION-->

<!ELEMENT Equation - - ((Title, Caption?)?, TextAlt?, (%mathcon;|Graphic))>
<!--      Title                                           (see common elems)-->
<!--      Caption                                         (see common elems)-->
<!--      TextAlt                                         (see common elems)-->
<!--      Graphic                                         (see graphic data)-->

<!--===================================================DYNAMIC ILLUSTRATIONS-->

<!ELEMENT Sound    - - ((Title, Caption?)?, TextAlt?, GraphicAlt?)>
<!ATTLIST Sound        %ident;
                       file     ENTITY   #IMPLIED>
```

Example C-3: (continued)

```
<!ELEMENT Video    - - ((Title, Caption?)?, TextAlt?, GraphicAlt?)>
<!ATTLIST Video        %ident;
                       file     ENTITY   #IMPLIED>
<!ELEMENT Applic   - - ((Title, Caption?)?, TextAlt?, GraphicAlt?)>
<!ATTLIST Applic       %ident;
                       file     ENTITY   #IMPLIED>
<!--      Title                                    (see common elems)-->
<!--      Caption                                  (see common elems)-->
<!--      TextAlt                                  (see common elems)-->
<!ELEMENT GraphicAlt
                   - - (Graphic)>
<!--      Graphic                                  (see graphic data)-->

<!--==================================================DIVISION-LIKE ELEMENTS-->

<!ELEMENT Procedure
                   - - (Title?, Task?, (Step|ProcExp)+)>
<!ATTLIST Procedure    %ident;
                       %L10Natt;>
<!ELEMENT Task     - O (%IUmix;)+>
<!ELEMENT Step     - O (%IUmix;)+>
<!ATTLIST Step         %ident;
                       perform  (OPT|REQ) "REQ">
<!ELEMENT ProcExp  - - (%divlikecon;)>

<!ELEMENT Scenario - - (Title?, (%divlikecon;))>
<!ATTLIST Scenario     %ident;
                       %L10Natt;>

<!ELEMENT SuppNote - - (Title, (%divlikecon;))>
<!ATTLIST SuppNote     %ident;>

<!--      Title                                    (see common elems)-->

<!--==========================================================BRIDGING TITLE-->

<!ELEMENT BrTitle  - O (%data5;)*>

<!--==========================================================EQUATION DATA-->

<!ELEMENT EqnData  - O (%mathcon;)>

<!--===========================================================GRAPHIC DATA-->

<!ELEMENT Graphic  - O EMPTY                       --file w/ graphic data-->
<!ATTLIST Graphic      not1     ENTITY   #IMPLIED
                       not2     ENTITY   #IMPLIED
                       not3     ENTITY   #IMPLIED
                       not4     ENTITY   #IMPLIED
                       not5     ENTITY   #IMPLIED>

<!--======================================================GENERAL KEY DATA-->

<!ELEMENT Acro     - O (#PCDATA)>
<!ELEMENT DocCit   - O ((%data3;)*) -(DocCit)>
<!ATTLIST DocCit       rdesc    IDREF    #IMPLIED>
<!ELEMENT Author   - O ((%data3;)*) -(Author)>
<!ELEMENT Term     - O ((%data4;)*) -(Term)>
<!ATTLIST Term         rglos    IDREF    #IMPLIED>
<!ELEMENT NewTerm  - O ((%data4;)*) -(NewTerm)>
<!ELEMENT Emph     - O (%data4;)*>
<!ELEMENT Phrase   - O (%data3;)*>
<!ATTLIST Phrase       lang     CDATA    #IMPLIED
                       %L10Natt;>
<!ELEMENT Quote    - O (%data1;)*>
<!ELEMENT Sub      - O (%data3;)*>
<!ELEMENT Super    - O (%data3;)*>
<!ELEMENT TM       - O ((%data4;)*) -(TM)>
<!ATTLIST TM           rattrib  IDREF    #IMPLIED>
<!ELEMENT Address  - O ((%data1;)*) -(Address)>
<!ELEMENT Date     - O (%datecon;)>
```

Example C-3: (continued)

```
<!ATTLIST Date          %dateatt;>
<!ELEMENT Time     - O ((%data3;)*) -(Time)>
<!ELEMENT Calendar
                   - O ((%data1;)*) -(Calendar)>
<!ELEMENT Phone    - O ((%data3;)*) -(Phone)>
<!ELEMENT Honorific
                   - O ((%data3;)*) -(Honorific)>
<!ELEMENT Unit     - O ((%data3;)*) -(Unit)>
<!ELEMENT PrName   - O ((%data3;)*) -(PrName)>
<!ELEMENT MiscData - O (%data4;)*>
<!ATTLIST MiscData     %descrip;>

<!--=====================================================TECHNICAL KEY DATA-->

<!ELEMENT String   - O (#PCDATA)>
<!ELEMENT Const    - O (#PCDATA)>
<!ELEMENT Number   - O (#PCDATA)>
<!ATTLIST Number       radix    (BIN|OCT|DEC|HEX) #REQUIRED>
<!ELEMENT Not      - O (#PCDATA)>
<!ELEMENT Key      - O (%data4;)*>
<!ELEMENT Win      - O (#PCDATA)>
<!ELEMENT Arg      - O (#PCDATA|%synels;)*>
<!ATTLIST Arg          rdesc    IDREF    #IMPLIED
                       rmodel   IDREF    #IMPLIED
                       rrpage   IDREF    #IMPLIED>
<!ELEMENT Kwd      - O (#PCDATA)>
<!ATTLIST Kwd          rmodel   IDREF    #IMPLIED
                       rrpage   IDREF    #IMPLIED>
<!ELEMENT Cmd      - O (#PCDATA)>
<!ATTLIST Cmd          rmodel   IDREF    #IMPLIED
                       rrpage   IDREF    #IMPLIED>
<!ELEMENT Input    - O (#PCDATA|Key|Var)*>
<!ELEMENT Output   - O (#PCDATA|Key|Var)*>
<!ELEMENT Var      - O (#PCDATA)>
<!ELEMENT MsgCode  - O (#PCDATA)>
<!ELEMENT Msg      - O (#PCDATA)>
<!ATTLIST Msg          lang     (%nlslist;) #IMPLIED>
<!ELEMENT Rtn      - O (#PCDATA)*>
<!ATTLIST Rtn          rmodel   IDREF    #IMPLIED
                       rrpage   IDREF    #IMPLIED>

<!--==================================================CROSS-REFERENCE LINKS-->

<!ELEMENT XRef     - O EMPTY>
<!ATTLIST XRef         rid      IDREFS   #REQUIRED   --pts to thing or ldesc-->

<!ELEMENT DocRef   - O EMPTY>
<!ATTLIST DocRef       rdoc     IDREFS   #REQUIRED          --pts to docdesc-->

<!ELEMENT RPageRef - O EMPTY>
<!ATTLIST RPageRef     rrpage   IDREFS   #REQUIRED     --pts to pg or rpdesc-->

<!--=========================================================FOOTNOTE LINKS-->

<!ELEMENT Footnote - O (%IUmix;)+ -(Footnote)>
<!ATTLIST Footnote     rid      IDREF    #CONREF>

<!--=======================================================OTHER DATA LINKS-->

<!ELEMENT Spot     - O EMPTY>
<!ATTLIST Spot         id       ID       #REQUIRED>
<!ELEMENT IdxSpot  - O (%idxentrycon;)>
<!ATTLIST IdxSpot      %ident;>
<!ELEMENT Link     - O (%data4;)*>
<!ATTLIST Link         rids     IDREFS   #REQUIRED>
```

C.2 SGML Declaration Needed for the Prototype

The SGML declaration that satisfies the prototype's requirements for quantities and capacities is listed in Example C-4. See the IAWG's SGML Declarations and Parsers position paper for more information.

Example C-4: SGML Declaration (`compdtd.decl`)

```
<!SGML   "ISO 8879:1986"

-- OSF and prototype extensions to Reference Concrete syntax are labeled  --

CHARSET

BASESET  "ISO 646-1983//CHARSET
          International Reference Version (IRV)//ESC 2/5 4/0"

DESCSET    0    9   UNUSED
           9    1   9        -- tab --
          10    1   10       -- lf --
          11    1   UNUSED
          12    1   12       -- ff --
          13    1   13       -- cr --
          14   18   UNUSED
          32   95   32       -- space --
         127    1   UNUSED

BASESET  "ISO Registration Number 100//CHARSET
          ECMA-94 Right-hand Part of Latin Alphabet Nr.1//ESC 2/8 4/2"

DESCSET 128   32   UNUSED
        160   96   160

CAPACITY SGMLREF
                                       -- all are OSF extensions --
         TOTALCAP  260000              -- prototype extension --
         ENTCAP     50000
         ENTCHCAP   50000
         ELEMCAP    50000
         GRPCAP    220000              -- prototype extension --
         EXGRPCAP   50000
         EXNMCAP    50000
         ATTCAP     50000
         ATTCHCAP   50000
         AVGRPCAP   50000
         NOTCAP     50000
         NOTCHCAP   50000
         IDCAP      50000
         IDREFCAP   50000
         MAPCAP     50000
         LKSETCAP   50000
         LKNMCAP    50000
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
BASESET  "ISO 646-1983//CHARSET
         International Reference Version (IRV)//ESC 2/5 4/0"
DESCSET  0        128       0
FUNCTION RE                10          -- lf --
         RS                13          -- cr --
         SPACE             32
         TAB      SEPCHAR   9
NAMING   LCNMSTRT    ""
         UCNMSTRT    ""
         LCNMCHAR    "&#45;&#46;"
         UCNMCHAR    "&#45;&#46;"
```

Example C-4: (continued)

```
         NAMECASE    GENERAL  YES
                     ENTITY   NO
DELIM    GENERAL     SGMLREF
         SHORTREF    NONE
NAMES    SGMLREF
QUANTITY SGMLREF
         ATTCNT   140            -- OSF extension --
         GRPCNT    50            -- prototype extension --
         GRPGTCNT 160            -- prototype extension --
         LITLEN  2048            -- OSF extension --
         NAMELEN   32            -- OSF extension --

FEATURES

MINIMIZE DATATAG  NO
         OMITTAG  YES
         RANK     NO
         SHORTTAG NO
LINK     SIMPLE   NO
         IMPLICIT NO
         EXPLICIT NO
OTHER    CONCUR   NO
         SUBDOC   NO
         FORMAL   YES

APPINFO  NONE >
```

Note the following differences from the position paper's SGML declaration:

- The literal following the SGML declaration opening was changed to ISO 8879:1986 from ISO 8879-1986 because ISO has changed its standard numbering. This change is documented in *The SGML Handbook*.
- The callout for the right half of the 8859 codepage was changed to match the BASESET naming requirements of one of the parsers used to validate the prototype DTD.
- The GRPCAP and TOTALCAP capacities were increased. The recommended design makes the DTD somewhat restrictive, causing it to have a significant number of content models with large groups. Given this quantity of groups and tokens, GRPCAP was increased to 220000. TOTALCAP was increased as well, to 260000, to accommodate the size of GRPCAP and the other capacity counts.
- The FORM function character was removed because later IAWG discussions revealed that this function could be accomplished more appropriately with a processing instruction.
- The GRPCNT quantity was increased to 50 because of the number of elements in a single group in some content models. This is because of the number of specialized elements which can occur in some instances. This could have been circumvented in the DTD by the arbitrary addition of grouping delimiters around the appropriate sets of element names. This would not change the meaning of the content model in any way, but it was decided to not use this means, so that the quantity used by the DTD would be more accurately reflected.
- The GRPGTCNT quantity was increased to 160 to allow more groups and tokens to be specified in a single content model, to accommodate the DTD's large content models. There is no easy circumvention for this or for the increase in GRPCAP because the only way to eliminate the number of groups or tokens is to simplify the content model.
- The OMITTAG feature was set to YES to allow omitted tags in line with the change in recommendation of the design group. Note that the absence of

SHORTTAG requires all attribute values supplied in document instances to be specified within literal delimiters.

In general, the need for increased capacities and quantities is due to the ‘‘width’’ of some of the prototype’s content models:

- All the contexts that use the IU1 and IU6 mixtures for their content models, because of the recommendation to restrict the occurrence of bridging titles
- The specialized message, bibliography, and glossary divisions, because of the recommendation to require the presence of at least one instance of their respective specialized lists
- The specialized preface divisions, because of the recommendation to control the occurrence of the specialized subdivision elements

The sections that discuss these recommendations mention some possible alternatives if system or environment limits prevent their implementation.

Glossary

Following are definitions of terms used in this document. See the main text for descriptions of the semantic roles of individual components.

application
: A software system, other than a validating parser, that manipulates a document instance; examples of applications are formatters, computer-aided translators, database loaders, and information search and retrieval engines.

  See also ‘‘validating application’’.

asynchronous elements
: Two separate elements that, taken together, form the ‘‘start tag’’ and ‘‘end tag’’ of a single semantic component.

augmentation
: A software process that populates portions of an SGML instance, producing a new SGML instance (conforming to the same DTD or to a different DTD) that has enriched content. For example, augmentation can be used to populate a table of contents for a document, or to fill in standard linking information between command names in text and their reference pages.

bridging title
: A label for a location within a division or division-like element where the discussion changes focus. It is similar to a division title, without actually labeling any other element. It provides organizational structure to help the reader navigate the discussion but does not provide hierarchical structure.

caption
: An extended title for an illustration. (In these recommendations, the title component, not the caption component, is the primary label for an illustration.)

careful reader
: If you got this far, congratulations!

CDS
: A computer dialog/state illustration.

CFCM
: The Committee for Common Man, an ad hoc effort to develop a DTD and processing applications for reference pages that have traditionally been handled with the `man` tools.

character data
: Individual characters of text. In this document, this term is used informally to mean PCDATA (parsed character data), rather than the ISO standard meaning of CDATA (character data that is essentially unparsed).

chip
: An instance of information in a document for which more precise markup was available in the interchange DTD, but such markup was not used.

See also ‘‘spike’’.

component

See ‘‘semantic component’’.

data

Low-level components in the information pool that typically are available in a flat mixture as peers to PCDATA, and that themselves typically have a flat content model. Examples are emphasized phrases, command names, and superscripts. Some kinds of specialized data unique to particular information units also exist.

See also ‘‘flat’’ and ‘‘information unit’’.

data integrity

A characteristic of a markup language that has encodings of semantic distinctions that are abstract and precise enough to anticipate a wide variety of applications for the information over time.

displayed

Set off from surrounding narrative data-level text; a presentational characteristic.

See also ‘‘inline’’.

division

A cohesive, labeled collection of information that can have internal organization and/or can appear in higher-level such organizations, to give context to its contained information. Divisions can be generalized, specialized, or miscellaneous.

division-like element

A cohesive, labeled collection of information that has no internal subdivision organization and that is arranged somewhere inside a division as part of a series of information units. Examples of division-like elements are procedures and scenarios.

See also ‘‘bridging title’’.

document model

The set of assumptions about how readers use documents of a particular type, as manifested in the DTD and any applications that present the document to a user.

The document model is not to be confused with the information units that encode models of computer language entities, such as command models.

document type

A class of documents having similar characteristics, for example, software user manuals. A DTD codifies the semantic components and organization of such that make a class of documents unique.

DTD

Document type definition; a document type encoded in the SGML language.

excluded, exclusion

Disallowed from a certain context. In most cases in this document, the term ‘‘exclusion’’ does not necessarily mean exclusions in the SGML sense.

flat

An element or context whose content specifies elements only at a single level below it; or, an element available in such a context. For example, a simple list contains one or more items; therefore, simple lists contain a flat series of items.

Sometimes the term is used to refer to contexts that disallow nested hierarchies of themselves (or similar elements) in particular; for example, division-like scenarios are said to have a flat structure because no subdivisions are allowed within them, and glossary entries are said to be flat because they cannot contain glossary lists.

See also ‘‘modular’’.

flow, flow of text
: The cohesive information order chosen by a writer that reveals a logical progression of thought. At different levels, this means different things: a paragraph contains an ordering of data, a division contains an ordering of information units, and a manual contains an ordering of chapters. The chosen orderings are generally retained in the output.

Some components exist outside a flow: for example, footnotes, division summaries, and all components in the metainformation container.

GCA
: Graphic Communications Association, based in Alexandria, Virginia. GCA makes available much useful information on SGML theory and practice.

generalized element
: An element, such as a D1-level division or an ordered list, that can be used to hold any kind of information of the writer’s choosing.

grid pattern
: A writing technique where several divisions or lists (for example) are used in series to document related topics, and a template of common subdivisions or list items is available within them. Only the relevant parts of the template are used for each topic, necessitating a model that allows single subdivisions or list items in their containing divisions or lists.

HyTime
: ISO standard 10744, Hypermedia/Time-Based Structuring Language. HyTime is a hyperdocument structuring language encoded in SGML form.

illustration
: Information whose data integrity is, to some degree, sensitive to the presentation of the information. Examples of illustrations are figures and sound clips.

information pool
: See ‘‘pool’’.

information unit (IU)
: High-level component in the information pool that typically has internal structure and culminates in data. Examples are paragraphs, command models, and tables.

See also ‘‘data’’.

inline
: Incorporated into the surrounding narrative data-level text, rather than being set off; a presentational characteristic.

See also ‘‘displayed’’.

interchange
: The sharing of source-form documents for the purpose of modifying and reusing them. It implies a negotiated data protocol for the documents and a set of processes for validation.

ISO

The International Organization for Standards.

IU

See ‘‘information unit’’.

key data

Data components that identify a small piece of information that readers and applications may want to ‘‘key on’’, for example, a command name. Key data is of two kinds: general and technical.

licensee

A receiver of OSF offerings and its associated documentation.

link

The association of an origin location or element with one or more target locations, elements, or external entities.

manual

A document type in the scope of this project; it is typically used to hold a large portion of the relevant information on a particular computer subject area, organized according to a traditional ‘‘book’’ document model. The logical type does not dictate that the presentation form be a printed book.

miscellaneous element

A division or other element that can hold any kind of information but requires its contents to be identified as if it were a specialized element.

See also ‘‘specialized element’’.

model

See ‘‘document model’’.

modular

Able to stand alone, without additional context. For example, reference pages and glossary entries are modular because they are assumed to contain everything the reader needs to know about the subject, or if not, they explicitly direct the reader to the other sources of information. Their containing list or division is not intended to provide additional essential information for understanding them.

See also ‘‘flat’’.

navigation

The process of finding desired information in a document. Tables of contents and indexes are highly stylized mechanisms for helping readers to navigate a manual; the ‘‘augmented navigation information’’ mentioned in this document refers to them. Other more free-form mechanisms are introductory paragraphs, bridging titles, and cross-references; these are created at the discretion of the writer.

NLS

Codes that indicate a software product’s ‘‘national language support’’ in its interface text. ISO standard 639 specifies codes that represent natural languages, and ISO standard 3166 specifies codes that represent territories or countries that use versions of these languages. OSF uses these two codes as part of its locale names in internationalization software; they are probably sufficient as the values of language attributes in the components for product ID, message text, and phrase in a different language.

origin

The location or element that is the starting point of a link.

PCDATA
: Character data that is parsed to locate markup (such as entity references).

PI
: An SGML processing instruction.

placeholder
: A mnemonic name standing for a variable value or series a values that the user or the system must supply.

pool
: An available group of information components. The information pool is the collection of all information units and data available within a document type. The information unit pool comprises the high-level elements in this group, and the data pool comprises the low-level elements in this group.

  See also ‘‘data’’ and ‘‘information unit’’.

presentation
: The characteristics of how a document’s information is rendered (visually or otherwise) for a user’s consumption, rather than what the information means.

presentation instance
: A document instance that has been presented to a user by means of a formatting application.

protocol
: A data form for use in document interchange that has been negotiated and agreed to by a set of interchange partners.

receiver
: An organization that accepts a source-form document in an interchange transaction. In OSF-related interchange, the initial receiver is OSF itself when technology suppliers send it documents or document fragments. The ultimate receivers are licensees, which get source-form documents along with the offerings.

sender
: An organization that transmits a source-form document in an interchange transaction. In OSF-related interchange, the initial senders are any technology suppliers that contribute documents or document fragments to the development of an OSF offering. The final sender is OSF itself, which transmits source-form documents to licensees along with the offerings.

SIL
: A segmented item list.

SML
: Semantic Markup Language, OSF’s structured `*roff`-based macros for documentation.

specialized element
: A division, such as a bibliography, or other element that can be used to hold only a particular kind of information.

  See also ‘‘miscellaneous element’’.

spike

An instance of a semantic distinction made in a document and its markup language that the interchange DTD does not make; markup that is relevant to some subset of suppliers and/or licensees.

See also ‘‘chip’’.

supplier

A contributor to the development of an OSF offering and its associated documentation.

Tag Abuse Syndrome

A case where a writer has applied the wrong semantic component to information because the writer either feels the markup is more convenient thatn the correct markup, or prefers the appearance that formatting applications apply to it.

target

A location, element, or document that is the ending point of a link.

TAS

See ‘‘Tag Abuse Syndrome’’.

TOC

A table of contents.

validating application

An application that checks for characteristics of interchanged documents that are not in conformance to the data protocol, but that a validating parser cannot catch. For example, marked sections are ‘‘legal’’ constructs in all SGML document instances; DTDs and SGML declarations cannot prohibit their use. If they are prohibited in interchanged documents, therefore, some application other than a parser must be used to warn of their presence in an interchanged document instance.

volume

A presentational fraction of a manual, as opposed to a logical subtree, that exists because of printing or binding constraints.