Publishing Solutions
InfoMaster Architectural Form Reference

February 13, 1994

Team Leader: Wayne L. Wohler, WOHLER at BLDVM1, t/l 263-0470
Eliot Kimber, DRMACRO at RALVM13, t/l 441-7091
Don Day, DAY at KGNVMY, t/l 695-8852
Simcha Gralla, SGRALLA at BLDVM1, t/l 263-0334
Mike Temple, TEMPLE at RCHVMW2, t/l 553-0682

Notice to reader:

This document is the first public draft of this material and as such is subject to revision. This document is not a commitment by IBM to provide any function or feature as a product or service at any time in the future.

About This Book

Who should read this book

This book is written for information development professionals and for programming professionals who have experience in supporting information development activities.

What you should already know

The reader must have knowledge and experience with IBM's BookMaster application. The reader must also have knowledge of Standard Generalized Markup Language (SGML) and its concepts and principles. Knowledge of HyTime would be very useful in understanding this document as well.

How this book is organized

Conventions used in this book

In general, the default BookMaster style will be used to present information that is marked up in accordance with BookMaster definitions. So, for example, citations marked up with :cit. will be set in italics. Some special conventions have been used:

- All element and attribute names will be set in small caps like this: ELEMENT.
- Many of the special terms used in this book are defined in the glossary found in the back of the book.
- SGML terms are used according to the intent of the ISO 8879 standard and are avoided for other concepts. This may cause confusion in areas where the SGML term's meaning is different than that used outside of the SGML community. There is one exception to this, the term **link** may be used to mean hyperlinks or cross-references while in SGML, it is a feature having to do with the association of processing attributes with elements.

Books in this Library

- *IBM ID Doc Development Work Book*

  This book is an overview of the entire IBM ID Doc definition project. It also contains development papers which describe issues that were addressed during the development effort.

- *InfoMaster Architectural Form Reference*

  This book contains the definitions of all the InfoMaster architectural forms that are the basis for the IBM ID Doc language design. It also contains some introductory material about architectural forms in general and InfoMaster architectural forms in particular, which is shares with the IBM ID Doc Development Work Book.

- *IBM ID Doc Language Reference*

© Copyright IBM Corp. 1992, 1994

This book contains the definitions for all the IBM ID Doc elements.

- *IBM ID Doc Style Specification Language Reference*

  This book contains the definition for the proposed IBM ID Doc style specification language. This is work in progress and may not be complete at the same time all the other books are distributed.

What else to read

What to read before reading this book

- *IBM Publishing Systems SGML Translator Creating an SGML Document for DCF Processing*, SC34-5076. Use this book to learn basic SGML concepts and terminology.
- *IBM Publishing Systems BookMaster BookMaster User's Guide*, SC34-5009. Use this book to become familiar with IBM's current BookMaster product offering.

Other books to help you

- *International Standard ISO 8879 — Standard Generalized Markup Language*. Use this book to become familiar with the SGML standard. An amendment to this standard has been adopted and is published separately and should be read at the same time as the standard.
- *International Standard ISO 10744 — Hypermedia/Time-based Structuring Language (HyTime)*. Use this book to become familiar with the HyTime standard.
- *The SGML Handbook* by Charles Goldfarb. Use this book for a friendlier introduction to SGML. It also contains a complete annotated standard and other information and documents of interest to those interested in SGML.
- *Practical SGML* by Eric van Herwijnen. This book is an overview of SGML and the processing systems that are available for SGML. It has some practical examples and chronicles Eric's experiences with SGML.

Contents

About This Book . . . ii
Who should read this book . . . ii
What you should already know . . . ii
How this book is organized . . . ii
Conventions used in this book . . . ii
Books in this Library . . . ii
What else to read . . . iii
  What to read before reading this book . . . iii
  Other books to help you . . . iii

InfoMaster Introduction . . . 1

InfoMaster Architecture Model . . . 2

InfoMaster Concepts . . . 4
Identification . . . 4
Semantic Class . . . 4
Semantic Property . . . 4
Descriptive Property . . . 5
Containers and Their Use . . . 5
  Content References . . . 5

InfoMaster Architectural Forms . . . 7
Overview . . . 7
  What is an Architectural Form? . . . 7
  How are Architectural Form's Used? . . . 7
  InfoMaster Functional Organization . . . 8
Markup Design Categories . . . 9
Conventions used in Architectural Form Definition . . . 11
  Naming Conventions . . . 12
  Other conventions used in the architectural forms . . . 12
Description of the attributes common to all architectural forms . . . 13
Architectural Forms Definitions . . . 15
  Caption . . . 16
  ConceptBridge . . . 17
  Container . . . 18
  ContainingDivision . . . 19
  CrossReference . . . 20
  DefiningList . . . 22
  Description . . . 24
  Display . . . 25
  Excerpt . . . 26
  HierarchicalDivision . . . 27
  HyperLink . . . 28
  InformationModule . . . 29
  LinkDescription . . . 30
  ModuleDescriptionItem . . . 31
  ModuleList . . . 32
  MultiMediaContainer . . . 33
  NameAndAddress . . . 34

© Copyright IBM Corp. 1992, 1994

ObjectContainer 35
Paragraph 36
Phrase 37
QueryLink 39
SinglePartList 40
SymbolicReference 41
Table 42
Term 45
Title 46

Conformance Levels with the InfoMaster Architectural Forms 47

Glossary 48

InfoMaster Introduction

The InfoMaster architecture provides a common base of document structure and semantics on which applications may be defined. It has been created to as part of the IBM ID Doc design effort but the scope of its application is not limited to IBM ID Doc. If fact, if it were, it would not meet its goal of providing this common base.

The semantics and services it supports are the following:

1. Source instance management

   The InfoMaster architecture provides the capabilities for including data, both conditionally and unconditionally. This may be done by referencing the data in one of several ways.

2. Property value management

   InfoMaster defines the concept of properties which may be associated with elements. These properties may be specified, referenced or inherited. InfoMaster does all of this processing.

3. Hyperlink management

   Hyperlinks and their management is a critical part of information management and InfoMaster supports the management of these links for both intra- and inter-document links.

4. Architectural form processing

   Applications may define their elements to conform to InfoMaster architectural forms. If they do, InfoMaster handles their processing for the application. This system will be general enough so that other architectures (like HyTime and Davenport) may use the same facilities for processing.

This section of the work book describes the InfoMaster architecture in detail, including requirements, an overview of the architectural forms and position papers on the issues addressed in the design of InfoMaster. For a complete definition of all the InfoMaster architectural forms see the *InfoMaster Architectural Form Reference*.

© Copyright IBM Corp. 1992, 1994

InfoMaster Architecture Model

The InfoMaster architecture is really a processing model which is intended to provide greater efficiency based on sharing processing code among multiple applications. It also emphasizes the similarities between applications by representing similar document structures with the same architectural forms. It also clearly identifies unique features of applications since these features do not conform to any InfoMaster architectural forms.

This section describes the processing anticipated by the architecture. Refer to Figure 1 while reading this section.

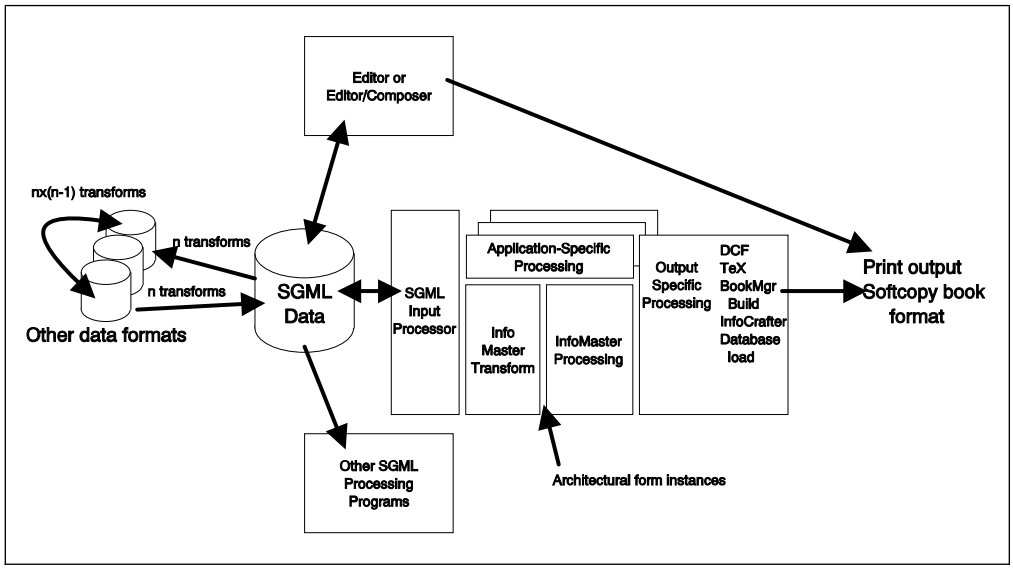


Figure 1. InfoMaster Architecture Diagram

The whole architecture is based on the use of SGML data conforming to the architecture as the interchange data form. This does not exclude the use of other data formats but the use of SGML data as an interchange form can reduce the number of transforms required significantly.

In addition, it is an open architecture. It is not required that the architecture be used to either create or process the data, as is shown. There will be significant processing efficiency gained if the architecture is used, as will be shown shortly, but it is not required.

The heart of the processing architecture is the series of processing steps shown from left to right. The first step is to parse the incoming SGML data and determine how further processing is to be accomplished. If the data conforms to the InfoMaster architectural form, the next step is to transform the SGML data to InfoMaster architectural form instances. This is shown by the block labeled "InfoMaster Transform". Conceptually at least, this block outputs the data using the architectural form names and attributes. This data is then processed by the "InfoMaster Processing" block and subsequently by the "Output-Specific Processing" block.

Note: The model is described and shown in terms of a sequential model since this is the model most readers will be familiar and comfortable with. Great pains

© Copyright IBM Corp. 1992, 1994

are being taken to assure that the architecture works with an interactive or "query" model as well.

It is recognized that the InfoMaster architectural forms cannot hope to cover all conceivable document structures. The architecture allows applications to provide their own processing for unique document structures. This processing is represented by the block labeled "Application-Specific Processing". Notice that this block has shadows. This represents the potential for multiple architectural forms being used in a single application, with the appropriate processing for each application being supplied. There must always be a "default" application-specific block to process data that is not marked as being processed by any architecture. Each of these "Application-Specific Processing" blocks then calls the "Output-Specific Processing" block for processing, just like the "InfoMaster Processing" block does.

The "Output-Specific Processing" block may contain many things. It could, for example, represent an intermediate form of data (like a BookMaster GML file, a TeX input file or an InfoCrafter input file) which is then processed by another program. It may consist of an API which is called to perform other functions directly. This is application-specific and not defined by the architecture.

It should be noted that in many respects this model mirrors the model that has been adopted by the DSSSL standards committee with its General Language Transformation Process (GLTP) on the front end with the Semantic Specific Processor (SSP) on the back end. This is intentional.

In any case, the main reason this architecture is valuable is that, for those application semantics that are common across all SGML data being processed, the architecture allows the processing to be shared easily. In addition, it provides a natural platform for translating from one application to another; in that case, the "Output-Specific Processing" block is outputting SGML data. Once a semantic has been invented and implemented as part of the architecture, it need not be implemented again. If changes to the semantics or the processing of the semantics are needed, this architecture minimizes the work required to make these changes. It accomplishes this by allowing the changes to one set of code which supports these processing semantics for all applications using the semantic.

InfoMaster Concepts

There are four concepts that are universal in InfoMaster architectural forms. They are specified by attributes in the architectural form. These four concepts are:

Identification

Every element may have a unique identifier for use by references, hyperlinks, content references or any other linking semantic that is devised. This identification is provided via an ID attribute. This ID attribute will not be declared using the SGML ID attribute declared value because of name scope problems when more than one document are aggregated (since each document could have identical IDs specified). The NAME attribute declared value will be used instead.

InfoMaster manages the IDs and references to these IDs for use by processing programs which need this information for their processing. InfoMaster can also manage uniqueness so that the rule that a single document must contain only unique IDs is enforced (for example).

Semantic Class

The semantic class of an element is a classification of the role of an element that InfoMaster understands and processes. This can be specified at any level and its value is inherited from an element's parent if it is not specified. The semantic classes that are defined:

Annotation indicates that the element is being used as supplementary material, not as inline content.

MetaInformation indicates that the element contains information about its first non-metadata ancestor. Meta-information is the classification or description of an element. Meta-information must not be treated as inline content of the element it describes.

Content This semantic class indicates that the element contains data which is part of the main business or purpose of the element.

These three semantic classes are mutually exclusive.

Semantic Property

understood and processed by InfoMaster. The knowledge of these semantics by InfoMaster is the key distinguishing feature between “semantic properties” and “descriptive properties” (whose definition follows). The following list gives all the semantic properties that are currently defined for InfoMaster:

definition indicates that an element type has the same attribute list definition as another element type. Instances of the other element type can inherit attribute values from instances of this element type. This is the InfoMaster implementation of the xDEF elements found in BookMaster that were so successfully used in that application.

© Copyright IBM Corp. 1992, 1994

enumeration indicates that the element is enumerated or represents enumerated data.

Descriptive Property

This is a property that InfoMaster carries for an application while it has no knowledge of the semantics of the property. It can act on the property values of elements by doing comparisons and existence tests. Examples of this type of property are:

- Language
- Revision level
- Version
- Process specific control information
- Qualification information

Containers and Their Use

Containers are a new concept that is being introduced in InfoMaster and IBM ID Doc. The idea is that a container contains a set of elements which are held within the container but not processed at the point at which they are entered in the document. The intent is that they be used by reference from other locations in the document. These containers may or may not contain a homogeneous set (that is a set where the top element in the container is the same for all elements in the container). This depends on the definition of the container in the application-defined DTD.

Use of data within containers is accomplished by reference and is conditioned by "properties". A property is a value that is held in an attribute on the element in the container. The processing system may or may not use the element from the container based on 1) the value of the property attribute and 2) the conditions the processing run is using. For example, one property is language. It is possible to put all the different language versions of a paragraph in a container and then refer to ALL of these different versions from the document. The processing system would then either use or ignore those versions that are not intended for inclusion. If the current processing run is only interested in English and French (for a Canadian publication) then German and Italian would not be included. This concept is very powerful and will be used for version control, revisions and multiple language support.

Content References

Content references are references which pull in content from a point that is referenced. This implements structure-oriented data reuse and in some respects mimics entities except that it is, by definition, limited to proper document subtrees. There are two levels of support of content references:

1. Content references to information contained in "containers" only
2. Content references to information found anywhere in the document (the element type of the referenced data must still be valid for the content reference)

This function is defined in the OBJECTREFERENCE architectural form, in MULTIMEDIACONTAINER.

The following example shows some of these concepts in an alternate language versions application. In this case, the alternate language versions are in the vicinity of the primary version.

```
<Paragraph ID=p20e
           RetrievalAlternatives='p20f p20g'
           Language=English>
This is the twentieth paragraph of the primary document.
<Paragraph ID=p20f Language=French>
Ici est de vingti&eg;me paragraphe ...
<Paragraph ID=p20g Language=German>
Hier ist der zwanzigst Paragraph ...
```

Notice that the Properties attribute was used to specify the Language property. Note that the ID attributes are the same IDs used to identify this element for any other purpose.

InfoMaster Architectural Forms

Overview

The InfoMaster architecture is an architecture that is used as a base for defining SGML applications. As we have seen, InfoMaster supports several important and useful functions for these applications. The importance of architectural form's is that they are the means that applications use to indicate to InfoMaster processing routines the elements that represent InfoMaster-supported semantics. This is accomplished using InfoMaster-defined attributes with values equal to special keywords. InfoMaster can then interpret the content of these attributes and take the appropriate actions. This allows many applications to exploit InfoMaster-provided function with a minimal intrusion into the definition of the applications themselves. The architectural forms themselves can be thought of as meta-DTD fragments which guide application writers in defining their applications. This concept comes from HyTime and will get a bit more introduction now.

What is an Architectural Form?

Architectural forms are the device that InfoMaster uses to identify elements in application DTDs that use InfoMaster semantics. Architectural forms are defined using the same syntax used by SGML to define element types and attibutes for those elements. In this case, they are meta-element definitions that state some rules about the structure, content and attributes of elements which conform to the InfoMaster architecture. These forms describe what is allowed in an application's DTD, in the same way that the DTD describes what is allowed in a document instance. These forms are used to guide the writers of DTDs in insuring that the elements that they are defining conform to the architecture. Unfortunately, an SGML parser cannot be used directly to do this validation. The reader should refer to the HyTime standard (ISO 10744), which has a section describing architectural forms in greater detail for more information on this topic.

How are Architectural Form's Used?

Architectural forms are models which codify the rules for writing the declarations for elements and attributes where these constructs must conform to the architectural forms. In particular, architectural form's specify what is required, what is optional, where order is important and where it is not. This information is encoded using the same keywords and syntax as a SGML declarations used in the DTD itself, supplemented where necessary with comments. The linkage between the DTD and the architectural form that is necessary for processing systems to process documents which conform to the architectural forms is provided by names and content of attributes. There are two key attributes which form the basis for these associations:

- INFOMASTER attribute
- INFOMASTERNAMES attribute

The following sections describe these two attributes in more detail.

© Copyright IBM Corp. 1992, 1994

InfoMaster Attribute

This attribute communicates to the processing system the name of the architectural form of InfoMaster that the element conforms to. This attribute is required on any element that intends to conform to an architectural form. The name of the attribute must be InfoMaster unless APPINFO in the SGML declaration for the document contains the InfoMaster keyword with a different character string following. In that case, that character string must be the name of the attribute which is used for this purpose.

InfoMasterNames

Having now identified the architectural form that the element conforms to, the element must have attributes which contain information that the architectural form requires and may have other attributes which the architectural form processor uses if they are defined. These attributes are known to the architectural form processor by their name. The element must define these attributes using the same names as the architectural form unless their names are mapped in an InfoMasterNames attribute. This attribute contains pairs of names, the first name of a pair is the element's attribute name and the second name in the pair is the architectural form's attribute name. With this mapping facility, the DTD now has the freedom to use attributes which will have the most meaning to the user while still mapping their semantics to those understood by the architectural form.

Specific Rules for Writing and Using Architectural Forms

Note to reader:

The material in this section will be supplemented with more detail on the rules regarding DTD conformance to architectural forms. Some of the questions that need to be answered are:

- What are the rules for each content model connector and occurrence indicator?
- What are the rules for each different default value keyword in attribute list declarations?
- Can a content model token in an architectural form content model be replaced by the content model of that token in the content model of the element declaration in an instance of the architectural form?
- How are elements that conform to multiple architectural forms from different architectures (like InfoMaster and HyTime) defined?

InfoMaster Functional Organization

Since InfoMaster is implemented as a base of required function with modules or “towers” of optional function, function is divided into three categories: base, general tower and specialized tower function. Base comprises those functions which in our judgement were required in even the simplest document preparation applications. General towers are sets of function that have general application but are not required in all applications. General towers may also be expensive to implement and therefore need to be segregated for business reasons as well. There is a tradeoff between general use and expense in defining elements as base or as a general tower. Specialized towers are those towers which are not general, they have very specific and specialized uses and may occur only in very specialized documents.

Markup Design Categories

The following table gives an overview of the markup categories that are to be designed.

The first entry is a brief description of the category. In some cases, this description includes bulleted lists with further breakdown of the category.

The second entry is a name that has been assigned to the architectural form defined for the particular category. These will be defined as the form is defined.

The third entry indicates whether the architectural form is included in the base or is a tower.

The fourth entry indicates the level of conformance required of a system to support the architectural form being described. For more information about levels of conformance, see "Conformance Levels with the InfoMaster Architectural Forms" on page 47 for more information about InfoMaster levels of conformance.

The fifth entry indicates whether the architectural form will be supported in the prototype system being planned.

The InfoMaster architectural forms are segmented into the base architectural forms and those which are found in towers. The following table lists those architectural forms that are found in the InfoMaster base definition. Also listed is whether they are going to be included in the prototype to be build when the definition phase is complete. By definition, the support for all these base architectural forms is required for a conforming InfoMaster application and their conformance level is 1 (based on the conformance levels described earlier in this document).

The following table describes a number of other markup categories that were derived from the markup category table given earlier and from other sources (such as CALS). This table shows only those categories which were considered as candidates to be towers in the InfoMaster architecture. The table shows which of these categories were adopted as architectural forms. It also reflects our plans for supporting the category in the prototype. The conformance level indicates how the specific markup category fits into the conformance and support model for InfoMaster.

Category description	**Arch form name**		
	Base/Tower	**Conformance**	**Prototype?**
Document structure			
Document hierarchy (FRONTM, BODY, ...)	ContainingDivision		
	Base	1	yes
InfoMaster Document Element	ContainingDivision		
	Base	1	yes
Information hierarchy (H1, H2, ...)	HierarchicalDivision		
	Base	1	yes
Content structures (General)			
Single part lists (OL, UL, SL, NOTEL)	SinglePartList		
	Base	1	yes

Category description	Arch form name		
	Base/Tower	**Conformance**	**Prototype?**
Paragraphs (including NOTE)	Paragraph		
	Base	1	yes
Definition lists	DefiningList		
• Glossaries • Parameter lists	Base	1	yes
Concept bridge	ConceptBridge		
	Base	1	no
Content structures (special)			
Question and Answer (not complete computer-based tutorial)	QuestionAndAnswer		
	Tower	3	no
Complete computer-based tutorial	ComputerBasedTutorial		
	Tower	3	no
Excerpts	Excerpt		
	Base	1	yes
Bibliographic information	Bibliography		
	Tower	3	no
Trouble shooting procedures (BookMaster MAP)	N/A		
	N/A	5	no
Dialogs (human-machine)	HumanMachineDialog		
	Tower	3	no
Syntax diagrams, meta-language markup	Syntax (not implemented as form in prototype)		
	Tower	3	yes
Parts assemblies	N/A		
	N/A	5	no
Schedules [This is supported via reference to artwork generated by scheduling software.]	N/A		
	Not supported		no
Addresses	Address (not implemented as form in prototype)		
	Base	1	yes
Tables			
Tables - regular NxM and nested for complex row definitions	Table		
	Base	1	yes (no nesting)
BookMaster tables (ARRANGE-specification based like TABLE)	N/A		
	N/A	5	no
CALS tables	N/A		
	N/A	5	no
Exhibits - Generalized			
Display [base]	Display		
	Base	1	yes
Line specific data (LINES, XMP, SCREEN, ...) is not a semantic processed by InfoMaster.	N/A		
	N/A	N/A	yes
Images (ARTWORK)	MultiMediaContainer		
• Image and vector graphic data • Video and other multi-media	Base	1	yes
Linking			

Category description	Arch form name		
	Base/Tower	**Conformance**	**Prototype?**
Navigation link (L, SPOT, ...)	CrossReference HyperLink		
• Link descriptions (ILINK?) • Non-text object reference	Base	1	yes
Containers	Container		
	Base	1	no
Key Data			
Phrases	Phrase		
	Base	1	yes
Trademark information	Trademark		
	Tower	3	no
Copyright information	N/A		
	N/A	5	yes
Letters and Correspondence			
Letters and Correspondence (distribution lists, letterhead and signature)	N/A		
	Tower	3	no
Modular Information (can be viewed as a mini-document)			
• General modular information • Messages and codes • Procedures • BookMaster modular information (LERS, MSGL and CODEL)	ModularInformation		
	Base	1	yes
MAN pages (built on modular information)	Tower		
	N/A	5	yes

Several items were dropped off of the list of possible architectural forms. The following list gives these items and the rationale for moving them out of the list.

- Qualification - this is a characteristic or property of elements, not a data classification itself.
- Multilingual document handling - this is a characteristic or property of elements, not a data classification itself.
- Hyphenation - this is a processing concern, not a data classification problem.
- Process specific control - this is pure processing.
- Revisions - this is a characteristic or property of elements, not a data classification itself.
- Mathematics and character graphics - these will be handled using the notation processing capability and do not reflect need architectural forms.

Conventions used in Architectural Form Definition

Naming Conventions

During the course of defining the architectural forms language, we must generate many names for elements, attributes, attribute values, notations and other items. This section contains the conventions that we will use to create these names.

1. All names are valid English words or the concatenation of multiple English words.
2. Names will not be truncated or abbreviated; a variant SGML syntax will be used with a name length significantly longer than the 8 character limit used in the reference concrete syntax.

   Comment on Davenport convention for attribute names:

   The Davenport group is going to keep attribute names under eight characters to avoid a forced remapping when dealing with the reference concrete syntax. We feel this was unnecessary since systems that process HyTime will almost certainly be able to handle names greater than 8 characters in length.

3. If multiple words are used, they will be typed whereever used with the initial letter of each word capitalized. If the name is a single English word, its initial letter will be capitalized.
4. All names will be nouns with multiple word names consisting of a noun and one or more adjectives. Multiple word names will be presented in the order normally used in spoken English. Where this is ambiguous, the order **adjective — noun** will be used.
5. There are some abbreviations that are so universal that it would be more confusing to not use them than to follow the above convention, ID is one that comes to mind. These will be presented using all uppercase letters.
6. Where appropriate, names used within standards like HyTime, Davenport and important public applications will be used to avoid excessive remapping and to emphasize their synergy with the standard or application. One example of this is HyTime's use of the name LINKEND rather than REFID.

Other conventions used in the architectural forms

The architectural form defintions will use the following comment convention in content models (I have used the terminology invented by the Design Group which wrote the OSF Documentation SIG Recommendations since it is appropriate and it exists.):

%InfoUnitPool This indicates that the content model must contain the paragraph-like and lower elements. Typically, this content model would be an element content model; that is, it will not allow parsed character data to be entered directly.

%DataPool; This indicates that the content model must contain phrase-like elements and parsed character data.

%InfoPool; This represents the combination of both the **%InfoUnitPool**; and the **%DataPool;**.

%MetaInfoPool; This is the pool of meta-information elements. There are several categories of meta-information:

- Address

- Human name
- Business position title
- Title of information
- Property definition container
- Date
- Enterprise name
- Document control
- Revision control
- Content description

Each of these categories may be represented by application-specific elements. The concept of semantic class is used to allow any architectural form take the role of a meta-information element. Because of this abstraction (any architectural form may represent meta-information), the %MetaInfoPool; parameter entity described here is an abstract concept that does not have a concrete definition in the architectural forms themselves. This parameter entity's value must be set by a particular application.

Notice that if exceptions are needed, these should also be included in the comment. In some cases, a specific parameter entity may be defined to represent the content of specific elements.

Recommendations on tag minimization for elements using the architectural form are included in the SGML element declarations for the architectural form.

Description of the attributes common to all architectural forms

The following example shows the attributes which are defined for all elements in the InfoMaster architectural forms. A description of each follows.

```
ID                NAME          #IMPLIED
PropertyReference
                  NAMES         #IMPLIED     -- Must contain an ID. --
Annotations       NAMES         #IMPLIED
                                     -- Must contain a list of IDs. --
LinkEnds          NAMES         #IMPLIED
                                     -- Must contain a list of IDs. --
ContentReference  NAMES         #CONREF      -- Must contain an ID. --
RefType           NAME          #REQUIRED
                               -- (AttName, ElementName+)+
                                  #ANY as an ElementName indicates that
                                  any element may be referenced.
                               --
PresentationAttributes
                  NAMES         #IMPLIED                -- AttName+ --
InstanceElementName
                  NAME          #IMPLIED             -- ElementName --
InfoMasterSemanticClass
                  (Annotation|Metadata|Content)
                                #IMPLIED
                          -- Contains one of the specified tokens. --
Properties        CDATA         #IMPLIED
                       -- List of descriptive property attributes. --
ApplicationClass  NAME          #IMPLIED
                        -- Application-defined class of element --
```

Attribute name	Description of attribute
ID	Identification for the element that may be used to support any of the linkage or other functions that require referencing of elements.
PropertyReference	Used to point to another element whose specified properties are to be used as the properties of this element. These properties are used in preference to inheriting the properties of this elements ancestors.
Annotations	Used to link to another element which is an annotation associated with this element. This allows an element to point to its annotation, as well as the case where the annotation element points to the element it annotates.
LinkEnds	Used to point to another element. Its semantics are defined by the application that uses this link.
ContentReference	Used to link to another element whose content is copied to the content of this location. Typically, the target of this type of link must be of the same element type as the element doing the linking.
RefType	Generic identifier of elements which can be referenced. For example, this attribute may be used to limit the element types which may be referenced using the CONTENTREFERENCE attribute. This is specified using the following syntax: **(***attname***,** *elementname*+**)**+ #ANY is valid as an *elementname*, indicating that any element may be referenced.

PresentationAttributes List of presentation attributes. All non-ID, ID reference and presentation attributes are properties.

InstanceElementName Used in the architectural form instance to remember the name of the element from the SGML instance, before it was transformed.

InfoMasterSemanticClass Used to specify the semantic class of the element. The semantic class must be set to one of the three following types: Annotation, MetaInformation or Content. See “Semantic Class” on page 4 for more information on semantic classes. This attribute must be defined as #FIXED in a concrete DTD for elements with a fixed semantic class.

Properties This attribute lists the attributes of an element which represent descriptive properties. Since InfoMaster has no *a priori* knowledge of descriptive properties, they must be specified here for InfoMaster to process them properly. Multiple attributes on a single element may be named.

ApplicationClass This attribute is used by the application to identify the class of the element. Typically, the value of this attribute is fixed in the DTD, if it is used.

Architectural Forms Definitions

Caption 16
ConceptBridge 17
Container 18
ContainingDivision 19
CrossReference 20
DefiningList 22
Description 24
Display 25
Excerpt 26
HierarchicalDivision 27
HyperLink 28
InformationModule 29
LinkDescription 30
ModuleDescriptionItem 31
ModuleList 32
MultiMediaContainer 33
NameAndAddress 34
ObjectContainer 35
Paragraph 36
Phrase 37
QueryLink 39
SinglePartList 40
SymbolicReference 41
Table 42
Term 45
Title 46

Caption

Purpose

This architectural form defines a caption to a display or other formal presentation.

Description

A caption is a short phrase describing the content of a formal presentation within a document. This phrase is then used in tables of content and in cross references to the document construct containing the caption. The caption is also presented with the construct itself. The existance of a caption is used in BookMaster to determine if a construct is eligable to be referenced.

SGML Definition

```
<!element Caption - O (%DataPool;)* >
<!attlist Caption
                InfoMaster      NAME   #FIXED  "Caption"
                %CommonAttributes;
>
```

ConceptBridge

Purpose

Concept bridge is a concept that came out of the work of the Design subgroup of the Information Architecture Working Group (IAWG) of the Documentation Special Interest Group (DocSIG) of the Open Software Foundation (OSF). The idea is that within text, especially in magazines but in many other document types as well, there are structures that look like a title of a division (or section) but aren't. The key is that this title is not exclusively associated with the division that follows it but rather associates or “bridges” two adjacent sets of text. It does not share the semantics that one often associates with divisions and their titles; it is not included in the table of contents, the data which follows it does not stand alone with this “title” and should not be retrieved by itself in a electronic review application. Its role is to ease a transition for the reader from the old topic to the new topic and perhaps point out the relationship. This is all shown in an eye-catching way that resembles a division title.

The concept bridge does bridge or link the data which precedes and follows it. In this sense, it is a HyTime “ilink” with two anchors, which by default are the peer elements preceding and following the concept bridge element itself. This concept can then be generalized to a multiple link end element which associates all the link ends (they need not be sequential).

Description

The BridgingTitle form has an ID attribute to allow the title to be referred to by other, referencing, elements. The BridgingTitle form also has a properties attribute which allows support for multiple languages, versions and revisions, etc.

SGML Definition

```
<!element ConceptBridge - O ((%MetaInfoPool;)*, (%InfoPool;)*) >
<!attlist ConceptBridge
                HyTime          NAME    #FIXED  "ilink"
                InfoMaster      NAME    #FIXED  "ConceptBridge"
                AnchorRoles     CDATA   #FIXED  "link1 link2"
                %CommonAttributes;
>
```

Container

Purpose

Containers contain elements that are only used by reference. Elements that are contained are not processed at the point they occur in the source but are held for later use by reference, for example, via retrieval alternate reference from another element.

Description

Containers may contain any type of element. It is up to the application designer to determine content restrictions. Containers have two required processing semantics:

1. Elements contained are not processed immediately (or rather, the processing application is not informed of their existance) but are held for later use by reference.
2. Annotation elements are not implicitly linked to container elements. However, container elements may be explicitly linked to annotation elements. This enables the containment of annotations by containers without generating unwanted or unexpected assocations between annotations and container elements.

SGML Definition

```
<!ELEMENT  Container        - - ANY           -- Can contain anything -->
<!ATTLIST  Container
    InfoMaster       NAMED        #FIXED    "Container"
    %CommonAttributes;
>
```

ContainingDivision

Purpose

Containing divisions define portions of a document with a specific logical function but no defined hierarchical role. In other words, containing divisions do not influence the logical hierarchy defined by hierarchical divisions as distinct from the containment hierarchy as defined by the markup itself. The document element is always a containing division. Containing divisions are distinguished from paragraphs by being able to contain hierarchical divisions.

Description

There are no processing semantics associated with containing divisions except that containing divisions may not affect the logical hierarchy defined by hierarchical divisions. Containing divisions may or may not have a title associated with them. If so, the title is part of the division metadata.

SGML Definition

```
<!ELEMENT  ContainingDivision
                          - O
                     ((%MetaInfoPool;)*, Container*,
                      (ContainingDivision ] HierarchicalDivision
                      )*
                     )
>
<!ATTLIST  ContainingDivision
    InfoMaster       NAME         #FIXED    "ContainingDivision"
    %CommonAttributes;
>
```

CrossReference

Purpose

Creates a reference to another element, typically for the purpose of generating a hypertext link to that element where the text of the link is generated from the content of the element linked to.

All cross references are specialized hypertext links. Cross references, traditionally, are characterized by generating the reference text, as compared to explicit link phrases, where the reference text is defined by the author. SGML applications could use the same element for these two basic forms of reference, distinguishing between these uses with a CONREF attribute. But, this seems to us to be confusing. It also forces the coding of a CONREF attribute when the cross-reference behavior is desired (since an attribute must be coded if there is no content). Finally, explicit links are potentially complex enough to warrant their own architectural form. This form is called the HyperLink architectural form, it is described in “HyperLink” on page 28.

Description

The cross reference element conforms to the HyTime contextual link (clink) architectural form. Linkage can either be direct when the REFID= value is that of a normal element, or indirectly when the REFID= value is that of a LinkDescription element.

This form enables two methods of identifying links:

1. Using REFID alone

   This is the most common method. It points to another ID found within the same document as the reference. Notice that since it is defined as NAME, the application must validate that the target ID exists. The reason for defining this attribute as type NAME will be clarified next.

2. Using REFID with DOCID

   This method allows the author to specify the ID found in another document. The document within which the ID is found is indicated by the DOCID attribute. Since the DOCID attribute is found on the document element of the other document, this ID may not be in the name space of the current document and thus DOCID must have the type NAME rather than IDREF. The application will have to validate that the ID is encoded correctly. Since the REFID specification is in the ID name space of the target document, not this document, it must also have type NAME.

SGML Definition

```
<!ELEMENT  CrossReference   - O EMPTY >
<!ATTLIST  CrossReference
    InfoMaster       NAME         #FIXED    "CrossReference"
    HyTime           NAME         #FIXED    "clink"
    reftype          CDATA        #FIXED    "#ALL #ANY"
    reflevel         CDATA        #IMPLIED
    refrange         CDATA        #FIXED    "#ALL I"
    HyNames          CDATA        #FIXED    "RefID linkend
                                             DocID docorsub"
                -- docorsub exists only on the nameloc element today. --
```

```
    RefID           NAME        #REQUIRED -- ID of target --
    DocID           NAME        #IMPLIED -- ID of document
                                            containing target --

    %CommonAttributes;              -- excluding LinkEnds,
                                       ContentReference, RefType --
>
```

DefiningList

Purpose
Contains a set of terms and their associated definitions.

Description
A defining list contains a set of terms. Each term is associated with one or more definitions. There are no processing semantics associated with defining lists except that the association between a term and its definition(s) is preserved as an implicit hypertext link. InfoMaster does not provide any sorting or sequencing semantics for defining lists.

Terms and definitions can also be contained within Term containers.

The definition block provides a mechanism for organizing terms within a list, for example, into letters of the alphabet in a glossary list.

The nameloc-conforming element allows the author to generate a link to terms and their definitions that are distributed and use them in the context of the DefiningList. If the nameloc-conforming element is used, no textual content is allowed (although metainformation is allowed). If textual content is provided, then no query will be executed. If neither a nameloc-conforming element or text content is specified, the application will imply some processing based on the element in question. This will usually involve the execution of an application-defined query of some sort.

SGML Definition
```
<!ENTITY % DefiningListMetaData
    "Title?, TermHeading?, DefinitionHeading?, (%MetaInfoPool;)*"
>

<!ELEMENT  DefiningList      - -
                     ((%DefiningListMetaData;),
                      (nameloc? ]
                       (Term ] ConceptBridge ] DefinitionBlock)+
                     ))
>
<!ATTLIST DefiningList
    InfoMaster       NAME         #FIXED    "DefiningList"
    %CommonAttributes;
>
<!-- nameloc is the HyTime nameloc architectural form -->

<!ELEMENT  TermHeading       - O (%DataPool;) >
<!ATTLIST TermHeading
    InfoMaster       NAME         #FIXED    "TermHeading"
    %CommonAttributes;
>

<!ELEMENT  DefinitionHeading - O (%DataPool;) >
<!ATTLIST DefinitionHeading
    InfoMaster       NAME         #FIXED    "DefinitionHeading"
    %CommonAttributes;
>
```

```
<!ELEMENT  DefinitionBlock   - O ((%MetaInfoPool;)*, Term*) >
<!ATTLIST DefinitionBlock
    InfoMaster       NAME         #FIXED    "DefinitionBlock"
    %CommonAttributes;
>
```

Description

Purpose

This architectural form defines the description of a display or other formal presentation.

Description

A descruiption is a paragraph describing the content of a formal presentation within a document. The description is presented with the construct itself, often following the caption.

SGML Definition

```
<!element Description - O (%InfoUnitPool;)* >
<!attlist Description
                InfoMaster      NAME   #FIXED  "Description"
                %CommonAttributes;
>
```

Display

Purpose

This architectural form defines a display.

Description

A display is a construct used in documentation to set off a set of data from the data that surrounds it. Figures are a form of display, as are DF or display formulas from BookMaster. In order to be numbered, referenced or in other ways set off, data must be contained in a display. For example, formula data that is not contained in a display is flowed with the data that surrounds it, resembling a phrase in that context.

SGML Definition

```
<!entity  DisplayMetaInfo
                    "Title?, Caption?, Description?,
                     (%MetaInfoPool;)*" >

<!element Display - - ((%DisplayMetaInfo;),
                       (DisplaySegment*|(%InfoUnitPool;)*) >
<!attlist Display
                InfoMaster      NAME   #FIXED  "Display"

                Class           NAME   #REQUIRED
                        -- Class of display being defined:
                                Figure
                        --
                %CommonAttributes;
>

<!element DisplaySegment - O (%InfoUnitPool;)* >
<!attlist DisplaySegment
                InfoMaster      NAME   #FIXED  "DisplaySegment"
                %CommonAttributes;
>
```

Excerpt

Purpose

Excerpts are passages of text that are extracted from other sources for inclusion in the current work. In BookMaster, there are two elements which are used to represent this information, Q and LQ. This stands for quotation and long quotation, respectively.

The Q element is formatted as an in-line quotation, that is it is flowed with the surrounding text. This element is often used to simply enclose text within typographic quotation marks and contains a special term or a trite phrase rather than a true extraction from another work.

The LQ element gets more formal treatment in its formatting, it is typically presented with indented margins and may contain multiple paragraphs of text. It may also contain other data besides text. It always represents data extracted from another source.

Description

A single form will be used to contain excerpts which are flowed and those which are to be given more formal treatment in the presentation. This form will be used to represent data extracted from another source, not general data which needs to be presented with quotation marks (if supported, this type of presentation differentiation will be possible using the PHRASE architectural form).

The key aspect of this form is the tie it will require to a bibliographic entry which identifies the source of the excerpt. A complete description of the bibliographic entries is beyond the scope of this description but these entries will be referenced using an ID from this form. Other features of this form are an ID for referencing and a properties attribute to support multi-language, version and revision information and other properties to be determined.

Note that it may be desirable to directly reference the excerpted material, rather than copying it. This is done using a hyperlink in this architectural form's content.

SGML Definition

The following lines show the SGML encoding for the excerpt architectural form:

```
<!element Excerpt  - - ((%MetaInfoPool;)*,
                        (InformationModule)?,
                                    -- Its Class attribute must be
                                       specified as Class=Bibliography --
                        (%InfoPool;)*) >
<!attlist Excerpt
                InfoMaster      NAME    #FIXED  "Excerpt"
                BiblioID        REFID   #REQUIRED
                       -- Allows chaining of BiblioData with
                          union of data all data in the chain.
                       --
                %CommonAttributes;
>
```

HierarchicalDivision

Purpose

Defines the logical hierarchy of the document as distinct from the containment hierarchy of the document markup. Hierarchical divisions are always at a lower level than any parent hierarchical divisions above them, regardless of any intervening containing divisions.

Description

Hierarchical divisions have only one required processing semantic, that the logical hierarchy must be incremented by at least one for each level of nesting of hierarchical divisions. The division title, if there is one, is part of the division metadata. Hierarchical divisions can either represent absolute positions in the logical hierarchy (e.g., HeadLevel1) or relative position (e.g., Division element).

SGML Definition

```
<!ELEMENT  HierarchicalDivision
                            - O
                     ((%MetaInfoPool;)*, Container*,
                      (ContainingDivision ] HierarchicalDivision ]
                       (%InfoPool;)*
                      )*
                     )
>
<!ATTLIST  HierarchicalDivision
    InfoMaster       NAME         #FIXED    "HierarchicalDivision"
    %CommonAttributes;
>
```

HyperLink

Purpose

Defines an explicit link from a point in the document to another element. The HyperLink element can either contain the link phrase or be a cross reference link.

Description

When the LinkPhrase=automatic attribute is coded, the link phrase is generated by reference as for CrossReference elements. In all other ways HyperLink is identical to the CrossReference form.

SGML Definition

```
<!ELEMENT  HyperLink        - O (%DataPool;)* >
<!ATTLIST  HyperLink
    InfoMaster       NAME         #FIXED    "HyperLink"
    HyTime           NAME         #FIXED    "clink"
    reftype          CDATA        #FIXED    "#ALL #ANY"
    reflevel         CDATA        #IMPLIED
    refrange         CDATA        #FIXED    "#ALL I"
    HyNames          CDATA        #FIXED    "RefID linkend
                                             DocID docorsub"

    LinkPhrase       (Automatic)  #CONREF
    RefID            NAME         #REQUIRED           -- ID of target --
    DocID            NAME         #IMPLIED
                                   -- ID of document containing target--

    %CommonAttributes;                -- excluding LinkEnds, RefType --
>
```

InformationModule

Purpose

Provides a standard framework for structuring information consisting of a repeated set of elements, for example MAN pages, where each module describes a single object (e.g., command reference, message manuals, procedure documents).

Description

A single module consists of a name, description, and other possible metadata elements, followed by one or more divisions, called “module description items.” There are no required semantics for either the ModularInformation form or the ModuleDescriptionItem form. Applications may choose to treat them as divisions, either containing divisions or hierarchical divisions. They could also be treated as rows and columns in a table or fields in a structured data record.

Modules can be contained in a module list.

SGML Definition

```
<!ELEMENT  InformationModule
                            - O (ModuleMetaData, ModuleDescriptionItem*)
>
<!ATTLIST  InformationModule
    InfoMaster       NAME         #FIXED    "InformationModule"
    %CommonAttributes;
>

<!ELEMENT  ModuleMetaData   O O
                     (ModuleName, ModuleDescription?, (%MetaInfoPool;)*)
>
<!ATTLIST  ModuleMetaData
    InfoMaster       NAME         #FIXED    "ModuleMetaData"
    %CommonAttributes;
>

<!ELEMENT  ModuleName       O O (%DataPool;)* >
<!ATTLIST  ModuleName
    InfoMaster       NAME         #FIXED    "ModuleName"
    %CommonAttributes;
>

<!ELEMENT  ModuleDescription
                            O O (%DataPool;)*
>
<!ATTLIST  ModuleDescription
    InfoMaster       NAME         #FIXED    "ModuleDescription"
    %CommonAttributes;
>
```

LinkDescription

Purpose

Describes the target anchor of a link using indirection.

Description

LinkDescription conforms to the named address location form of HyTime (8.3.2.1) and thus contains either a namelist element, which contains a list of element IDs or a list of entity names, or a name list query element, which contains a query in some query language that is resolved to a list of element IDs or entity names. The element ID pointers may in turn point to arbitrary strings of characters using HyTime facilities.

LinkDescription also contains an optional metadata element that contains a description of the link target or purpose of the link.

Link descriptions can themselves refer to other link descriptions.

SGML Definition

```
<!ELEMENT  LinkDescription  - O ((%MetaInfoPool;)*, nameloc)* >
<!ATTLIST  LinkDescription
    InfoMaster       NAME         #FIXED    "LinkDescription"
    %CommonAttributes;
    %multloc.atts;  -- HyTime 8.2.2 --
    %spanloc.atts;  -- HyTime 8.2.3 --
>
```

ModuleDescriptionItem

Purpose

Identifies a standard section of an InformationModule.

Description

Module description items are assigned class values defined by the application designer. The class value can be defined by either the generic identifier of the concrete module description item element or via a CLASS= attribute. In either case the CLASS= attribute is required.

SGML Definition

```
<!ELEMENT  ModuleDescriptionItem
                            O O
                     ((%MetaInfoPool;)*, (%InfoPool;)*)
>
<!ATTLIST  ModuleDescriptionItem
    InfoMaster       NAME         #FIXED    "ModuleDescriptionItem"
    ApplicationClass NAME         #REQUIRED
                       -- The list of classes either defined in the
                          application-specific DTD or is defined as
                          #FIXED if the GI determines the class value. --
    %CommonAttributes;                -- Excluding ApplicationClass --
>
```

ModuleList

Purpose

This form identifies a collection of InformationModules.

Description

SGML Definition

```
<!ELEMENT  ModuleList
                              - -
                     (InformationModule)+
>
<!ATTLIST  ModuleList
    InfoMaster       NAME         #FIXED    "ModuleList"
    %CommonAttributes;                  -- Excluding ApplicationClass --
>
```

MultiMediaContainer

Purpose

This architectural form defines the multi-media container which is used to contain all non-textual content of the document.

Description

The multi-media container is used to contain all non-textual material to be included in the document. An object may be included directly, using the object container, or by reference. In addition to the object container (which may contain textual, even SGML markup) definitions of the object, the multi-media container requires a textual alternative be provided for visually-impaired readers.

The multi-media container is a logical container representing one logical item, which is composed of two physical pieces of data: the text alternative and the multi-media data itself (which is either contained or referenced).

SGML Definition

```
<!element MultiMediaContainer
                     - - ((%MetaInfoPool;)*,
                           (ObjectContainer]
                            HyperLink]
                            CrossReference)*,
                          TextAlternative) >
<!attlist MultiMediaContainer
                InfoMaster NAME #FIXED "MultiMediaContainer"
                %CommonAttributes;
>

<!element TextAlternative
                     - - (%InfoUnitPool;)*
>
<!attlist TextAlternative
                InfoMaster NAME #FIXED "TextAlternative"
>
```

NameAndAddress

Purpose

The NameAndAddress form identifies name and address information.

Description

This form contains both a name and address. The semantic associated with this information is determined by the element in the document type using this architectural form. For example, this form may contain author names and addresses, document owner (either person or enterprise names), etc. Applications can use the content of a NameAndAddress element to resolve do special name and address processing, including whatever checking or retrieval that may be appropriate.

The content of an element conforming to this form must be a name followed by an optional address. Both the name and the address may contain either a blocking element which contains the name or address in a line-specific format with or without content markup indicating the semantics of its content or an ordered set of elements indicating specific parts of the name or address. The specific parts of the name and address of interest to a particular application are left to the application to define.

SGML Definition

```
<!ELEMENT  NameAndAddress   - O (Name, Address?)
                                          -- Symbolic Reference --
>
<!ATTLIST  NameAndAddress
    InfoMaster       NAME         #FIXED    "NameAndAddress"
    %CommonAttributes;
>

<!ELEMENT  Name             - O (NameBlock? ] (%DataPool;)*) >
<!ATTLIST  Name
    InfoMaster       NAME         #FIXED    "Name"
    %CommonAttributes;
>

<!ELEMENT  NameBlock        - O (%DataPool;)* >
<!ATTLIST  Name
    InfoMaster       NAME         #FIXED    "NameBlock"
    %CommonAttributes;
>

<!ELEMENT  Address          - O (AddressBlock? ] (Name?, (%DataPool;)*))>
<!ATTLIST  Address
    InfoMaster       NAME         #FIXED    "Address"
    %CommonAttributes;
>

<!ELEMENT  AddressBlock     - O (%DataPool;)* >
<!ATTLIST  Name
    InfoMaster       NAME         #FIXED    "AddressBlock"
    %CommonAttributes;
>
```

ObjectContainer

Purpose

This architectural form defines the object container which is used to indicate the physical data (by containing the data or by refering to it) which represents a multi-media item.

Description

The object container may consist of several parts, some of which are included by reference and it may include itself for situations where objects are nested. The NOTATION attribute must be specified if the element contains the data and must not be specified if the OBJECT is specified (since the entity declaration will give the notation).

There is a semantic difference between HYPERLINK or CROSSREFERENCE pointing to a multi-media object and OBJECTCONTAINER pointing to the object. In the former case, it represents a link which may or may not be traversed, at the reader's option, while in the latter case, the data is to be considered “in-line”, even though it may be presented separately or at the reader's option for performance or to conserve other system resources.

SGML Definition

```
<!entity  % ObjectElements "" -- application-specific list-->
<!element ObjectContainer - - ((%MetaInfoPool;)*,
                               (%ObjectElements;)]
                               ObjectContainer]
                               HyperLink]
                               CrossReference)* >
<!attlist ObjectContainer
                InfoMaster NAME     #FIXED    "ObjectContainer"
                Object     ENTITY   #CONREF
                Notation   NOTATION #IMPLIED
                %CommonAttributes;
>
```

Paragraph

Purpose
Contains paragraphs or other paragraph-like information.

Description
Paragraphs contain sentences or other phrases. Paragraphs may have generated titles (for example, a note paragraph element). The primary semantic associated with paragraphs is that paragraph text is always flowed when presented. Paragraphs may be distinguished from containing divisions by the fact that they cannot contain hierarchical divisions.

SGML Definition
```
<!ELEMENT  Paragraph        - O ((%MetaInfoPool;)*, ParagraphBody) >
<!ATTLIST  Paragraph
    InfoMaster       NAME         #FIXED    "Paragraph"
    %CommonAttributes;
>

<!ELEMENT  ParagraphBody    O O (%InfoPool;)* >
<!ATTLIST  ParagraphBody
    InfoMaster       NAME         #FIXED    "ParagraphBody"
    %CommonAttributes;
>
```

Phrase

Purpose

This architectural form represents information found within paragraphs and other similar elements. This data would normally be simply parsed character data but for some reason, additional significance is associated with the data. Since this data is typically one or more words and less than a sentence it is called “phrase”. This form provides the data description capability that BookMaster provided using the highlighted phrase elements (HPx), PK, PV, HEX, BIN, CHAR, XPH, etc. and will be expanded to allow the markup of other kinds of information as well.

Description

This element is part of the **data pool** class of elements and is therefore allowed anywhere that parsed character data is permitted.

The Phrase architectural form is fully recursive with all elements of the data pool.

The different classes of highlighted phrases (HP0-9) found in BookMaster are all supported in Phrase but the use of the general classes should be reduced with the addition of more descriptive classes which should always be used in preference to the more general classes. In addition, the user may easily define additional classes in their markup language.

SGML Definition

```
<!element  Phrase       - - (%DataPool;)* >
<!attlist  Phrase
               InfoMaster      NAME    #FIXED  "Phrase"
               Class           NAME    #REQUIRED
                             --   Address              |
                                * BinaryData           |
                                * CharacterData        |
                                  ColloquialTerm       |
                                  CommandName          |
                                  DaemonName           |
                                * DecimalData          |
                                * ExamplePhrase        |
                                  Expletive            |
                                  FirstUseOfTerm       |
                                  ForeignLanguageTerm  |
                                  FunctionName         |
                                * HexidecimalData      |
                                  HighlightedPhrase0-9 |
                                  HumanName            |
                                  Jargon               |
                                  KeywordText          |
                                * MarkedDeletion       |
                                  MenuItem             |
                                * MessageVariable      |
                                * OctalData            |
                                * ProgrammingKeyword   |
                                * ProgrammingVariable  |
                                  Slang                |
                                * Subscript            |
                                * Superscript          |
                                  VarablePlaceholder
```

Phrase

```
* indicates the class was used in the BookMaster SGML def --
                %CommonAttributes; >
```

QueryLink

Purpose

Defines an architectural form which represents a link to the results of a query. It combines the link to the query result with the query specification itself.

Description

QueryLink-conforming elements can be used by applications to support general, author-entered, queries or to support application-defined queries, like TOC, FIGLIST from BookMaster. QueryLink elements may contain a query specification or text content (ignoring the metadata that may be present), but not both. If a query is specified, the element represents a link to the results of that query. The system may present the result of this in any way that is appropriate to the media being used. If text content is specified, no query is executed and the text content is used to represent the element in the presentation or processing. Notice that the content model does not allow the specification of both a query specification and text content. If neither a query or text content is specified, then the action is determined by application defaults, which will often involve executing a query.

This element was designed to implement such BookMaster elements as TOC, PTOC, FigList, etc. It may also be used for doing collections like glossaries, which can be collected from descriptions entered within the document text.

SGML Definition

```
<!ELEMENT  QueryLink         - O ((%MetaInfoPool;)?,
                                  (nameloc? ] (%InfoUnitPool;)*)
                                              -- Content may be empty --
>
<!ATTLIST  QueryLink
    InfoMaster       NAME         #FIXED    "QueryLink"
    %CommonAttributes;
>

<!-- nameloc is the HyTime nameloc architectural form -->
```

SinglePartList

Purpose

This architectural form describes single part lists.

Description

This architectural form describes single part lists. In BookMaster, SL, UL and OL are considered to be single part lists, in that they consist of only a single description per item in the list. This is distinguished from a defining list which has two descriptions per item in the list.

SGML Definition

```
<!element SinglePartList - - ((%MetaInfoPool;)*,
                              (ListItem|ListItemBlock|
                               ConceptBridge)* >
<!attlist SinglePartList
                InfoMaster      NAME    #FIXED  "SinglePartList"

                DingBat         CDATA   #IMPLIED
                %CommonAttributes;
>

<!element ListItemBlock  - - ((ListItem|ListItemBlock|
                               ConceptBridge)* >

<!attlist ListItemBlock
                InfoMaster      NAME    #FIXED  "ListItemBlock"
                %CommonAttributes;
>

<!element ListItem       - O (%InfoPool;)* >

<!attlist ListItem
                InfoMaster      NAME    #FIXED  "ListItem"
                %CommonAttributes;
>
```

SymbolicReference

Purpose

The SymbolicReference form identifies references that may be logical or symbolic, rather than literal references to other elements. Applications can use the content of a SymbolicReference element to resolve references into actual links using queries. There is no requirement that the element refered to by a SymbolicReference element actually exist.

Description

This form is used to describe elements which have a need to document conceptual as well as concrete “linkages.” In these cases, it is not desirable to force the existance of a target using IDREF. Also, the title itself is used to represent the information being linked and is appropriate as content.

The content of an element conforming to this form must be a title and only TITLEBODY textual content is used to determine equality. It is not required that a match be found. The element type that an element conforming to this form references must have TITLE in its content model at some level (it may be in the content model of a subelement). An element instance of this architectural form must have an attribute TARGETTYPE which names the element type or types which should be searched when attempting to find symbolic references. This attribute may take one or more names or may be equal to “#ANY.”

SGML Definition

```
<!ELEMENT  SymbolicReference
                              - O (Title)         -- Symbolic Reference --
>
<!ATTLIST  SymbolicReference
    InfoMaster       NAME         #FIXED    "SymbolicReference"
    TargetType       CDATA        #REQUIRED
    %CommonAttributes;
>
```

Table

Purpose

Tables and their definition and markup are a subject which can cause a great deal of heated debate. Virtually every SGML application has its own unique and sophisticated table markup. It is not likely that an architectural form will be invented that is capable of capturing all the intent of all these different table markup systems. Complex tables is NOT the subject of this architectural form. This form defines a simple n by m format table. It covers the DIRECT and TABDATA elements found in BookMaster.

Description

The most fundamental description of this table is the number of rows and columns. Like most sequentially processed tabular information, this form defines tables in a row-wise fashion. The number of columns in the table is given in an attribute, the number of rows is derived from the number of sibling rows found in the table.

The fundamental unit of information in these tables is the CELL. There is no provision for logical rearrangement of cells in the markup of the architectural form. This functionality is in the purview of more sophisticated and complex tables. It is possible that the presentation semantics will provide the capability of presenting the information in an order that is different from the logical order defined in the document source.

It is not the intent of this form to replicate or preserve the entry mechanisms used to enter either BookMaster directories or tabbed data. Its objective is to capture the intent of these two table formats.

Headings, information which defines the meaning of the data in the columns is specified at the start of the table. In addition, row titles may be specified for each row (this is intended to map from the DIRSUB element found in BookMaster).

The form as defined has a recursive definition for TABLEBODY. This allows the table to include subheadings and yet still maintains (within the content of a TABLEBODY) the relationship between the heading and the data to which it applies for easy retrieval and isolation. While this may violate the idea of being simple, it does support the BookMaster idea of DIRSUB with a more hierarchical and flexible structure.

The form also allows the use of TABLEBODY within the TABLECELL element. This allows more complex table forms to be created. TABLEBODY may contain the data to be included directly or may like all elements, pull the data from another location using a content reference. In addition, there is a symbolic linkage possible using the TABLEBODYREFERENCE element which works like the FRAGMENT and FRAGREF BookMaster elements.

At this time, none of the presentation information specified in InfoMaster markup is defined in the archtectural form. It isn't clear how style information should be coded in InfoMaster (this is under study) and this could impact dramatically how this information is included in the architectural forms.

SGML Definition

```
<!element  Table        - -     ((%MetaInfoPool;)*,
                                 TableHeading?,
                                 TableFooting?,
                                 TableBody+) >
<!attlist  Table
           InfoMaster      #FIXED          "Table"
           %CommonAttributes;
>

<!element  TableBody
                        O O     (TableBodyName?,
                                 TableHeading?,
                                 TableFooting?,
                                 (TableRow |
                                         TableBody)+) >
<!attlist  TableBody
           InfoMaster      #FIXED          "TableBody"
           %CommonAttributes;
>

<!element  TableBodyName
                        - O     CDATA -- symbolic name for table body-->
<!attlist  TableBodyName
           InfoMaster      #FIXED          "TableBodyName"
           %CommonAttributes;
>

<!element  TableHeading
                        - O     (TableRow+) >
<!attlist  TableHeading
           InfoMaster      #FIXED          "TableHeading"
           %CommonAttributes;
>

<!element  TableFooting
                        - O     (TableRow+) >
<!attlist  TableFooting
           InfoMaster      #FIXED          "TableFooting"
           %CommonAttributes;
>
<!element  TableRow
                        - O     (Title?,
                                      TableCell+) >
<!attlist  TableRow
           InfoMaster      #FIXED          "TableRow"
           %CommonAttributes;
>

<!element  TableCell
                        - O     ((%InfoPool;)* |
                                 TableBodyReference |
                                 TableBody) >
<!attlist  TableCell
           InfoMaster      #FIXED          "TableCell"
           %CommonAttributes;
>
```

Table

```
<!element  TableBodyReference
                          - O     CDATA -- symbolic name for table body-->
<!attlist  TableBodyReference
                InfoMaster      #FIXED          "TableBodyReference"
                %CommonAttributes;
>
```

Term

Purpose

Contains a single term and one or more definitions.

Description

The Term form is intended to provide a mechanism for placing terms and their definitions in arbitrary places in a document independent of how those terms may be collected and presented. For example, glossary terms may be defined at the place they are first used in a document and collected for later use in an automatically-generated glossary or collected in a collection element from which terms actually used in a document are retrieved.

SGML Definition

```
<!ELEMENT  Term              - O
                    ((%MetaInfoPool;)*, DefinedTerm, TermDefinition+) >
<!ATTLIST  Term
    InfoMaster       NAME         #FIXED    "Term"

    RefType          NAME         #FIXED    "Term"
    SeeAlso          NAMES        #IMPLIED
    SortKey          NAME         #IMPLIED
    %CommonAttributes;
>

<!ELEMENT  DefinedTerm       - O ((%InfoPool;)*)
>
<!ATTLIST  DefinedTerm
    InfoMaster       NAME         #FIXED    "DefinedTerm"

    SortKey          NAME         #IMPLIED
    %CommonAttributes;
>

<!ELEMENT  TermDefinition    - O (%InfoPool;)* >
<!ATTLIST  TermDefinition
    InfoMaster       NAME         #FIXED    "TermDefinition"
    %CommonAttributes;
>
```

Title

Purpose
This architectural form defines the title of a display or other formal presentation.

Description
A title is a phrase describing the content of a formal presentation within a document. The title is presented with the construct itself.

SGML Definition
```
<!element Title   O O (ShortTitle?, TitleBody) >
<!attlist Title InfoMaster      NAME   #FIXED  "Title"
                %CommonAttributes;
>

<!element ShortTitle
                  - - (%DataPool;)* >
<!attlist ShortTitle
                  InfoMaster      NAME   #FIXED  "ShortTitle"
                  %CommonAttributes;

<!element TitleBody
                  O O (%DataPool;)* >
<!attlist TitleBody
                  InfoMaster      NAME   #FIXED  "TitleBody"
                  %CommonAttributes;
>
```

Conformance Levels with the InfoMaster Architectural Forms

For the purposes of discussing InfoMaster and its support of various applications, we identified five levels of conformance with InfoMaster:

1. Conformance with and use of InfoMaster base architectural forms.

   To meet this level, the application element's content model and attribute definition must conform to the architectural form including the use of the architectural form identifying attribute. InfoMaster processing of the InfoMaster processing semantics is automatic. The architectural form referenced is in the base of InfoMaster.

2. Conformance with and use of InfoMaster tower architectural forms.

   To meet this level, the application element's content model and attribute definition must conform to the architectural form including the use of the architectural form identifying attribute. Transformation to the InfoMaster architectural form instance is automatic. The architectural form referenced is in a general tower of InfoMaster. This is the only difference between this level and level 1 conformance.

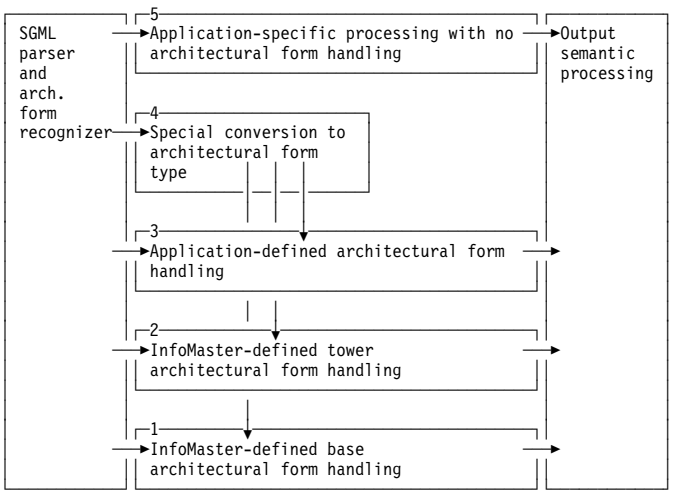
3. Conformance with architectural forms defined by the particular application being used

   This level of conformance permits the application designer to add architectural forms to the system, using the processing and automatic translation capability of the InfoMaster system for new architectural forms.

4. Element does not conform to any architectural form but is transformed to an architectural form instance using special transform code.

5. Element does not conform to any architectural form and is processed using custom code designed for that purpose.

The following graphic shows these levels and their relationships with each other.



© Copyright IBM Corp. 1992, 1994

Glossary

Since IBM ID Doc is an SGML application, SGML's terminology will be used whenever it applies and avoided when it does not. For the definition of an SGML term, refer to *ISO 8879:1986 — Standard Generalized Markup Language* or to *The SGML Handbook*.

application. An application is a collection of information class definitions. In SGML terms, an application is defined by a DTD which defines what information structures (elements) can and cannot be found in documents which conform to the application.

application domain. An application domain is a class of documents which is characterized by unique set information classes. They may also share the same application definition since this is a codification of the information classes which comprise the application domain.

architecture. An architecture is a definition or processing scheme which allows a single system to process documents from multiple application domains which are defined by different applications. Common processing is shared by all these domains or applications by abstracting to the information classes they share and processing each information class with the same code.

architectural forms. Architectural forms are a concept introduced by the HyTime standard. These forms are meta-element declarations and meta-attribute declarations. That means that they guide the writer of DTDs in writing DTDs by indicating how elements and attributes must be coded. If a DTD follows the rules indicated by these forms, it may be processed by systems designed to process data which conforms to the forms. That is the purpose of the definition of these forms, to codify limits so that processing systems may process data.

These architectural forms are the codification of the rules of information classes that characterize applications and therefore, application domains.

BookMaster. BookMaster is a product sold by IBM which defines a markup language for documents and then processes documents which use this language using Document Composition Facility (DCF). As used in this document, BookMaster may also mean only the markup language definition (exclusive of the processsing system).

character entity. The term character entity is used to describe an entity which represents a single character. Typically, this character is not representable in the document character set or at least cannot be entered on the keyboard that the author is using to edit the document. The replacement text of a character entity is often SDATA, since representation of a character not in the document character set is inherently system-dependent.

control word. This is the term used to describe the character sequences used by DCF to control its processing of document data. There are a large number of these control sequences defined by DCF. Typically, these control sequences start with a period which must be found in column one of an input line in the document data being processed.

cross-reference. A cross-reference is a hyperlink between two locations. It is unidirectional and typically causes generated text to be inserted by processing programs at the source point of the hyperlink. This generated text gives information about the information found at the other end of the hyperlink.

DCF. See “Document Composition Facility”.

document character set. The document character set is the collection

Document Composition Facility. Document Composition Facility (DCF) is a program product sold by IBM which takes document data marked up with special controls (called “control words”) and with generic markup called GML to produce hardcopy for a variety of printers.

Document Style Semantics Specification Language. This is the product of some work in progress in the international standards community which will result in an international standard. The standard will codify document formatting semantics. It will do this in the context of an architecture which can then be used to codify many other document processing semantics. Of particular interest is its processing model, which is used in the InfoMaster and IBM ID Doc processing models.

DSSSL. See “Document Style Semantics Specification Language”.

document type declaration. The document type declaration (DTD) is the SGML construct where all the elements, attributes and other markup constructs that are used in the document are declared. It is part of the prolog of an SGML document.

DTD. See “document type declaration”.

element. Element is an SGML concept. It contains the information associated with an information class. It may be further subdivided into other elements. It consists of a start tag (including any attributes), an end tag and content.

© Copyright IBM Corp. 1992, 1994

element type. Element type is the term used to refer to an element's definition, its name, its content model, its attributes, rather than a particular instance or usage of the element in a document. GD is an element type. This paragraph is found within an instance of a GD.

entity. An entity is an SGML construct which represents several ideas:

- It may be like a macro in the DTD, this type of entity is called a “parameter entity”.
- It may be like a GML symbol, with references replaced by a character string.
- It may define an external file containing SGML data. In this case, the reference is like the .im DCF control word, which references external GML data (only in SGML it is used to reference SGML data).
- It may define an external non-SGML file. In this case, the reference is like the .SI or .PO DCF control words or the NAME attribute of the ARTWORK BookMaster tag.

Entities are always defined in the prolog of the document.

entity reference. An entity reference is an SGML construct which is replaced by the information defined for the entity. The nature of the information which replaces the reference is dependent on the type of entity was defined.

formal public identifier. An FPI is a form of an SGML public identifier which conforms to the rules for formal public identifiers in the SGML standard. It is a system-independent way of refering to files from within an SGML document. Many SGML systems provide a means for converting these identifiers into a system identifier that the system processing the data can use to access data directly.

FPI. See “formal public identfier”.

generic identifier. A generic identifier is the name of an element type.

Generic Markup Language. GML is a generic markup scheme supported by Document Composition Facility. This scheme includes a tag (which may be a start or end tag), either of which may include “attributes” which further describe the data being “tagged”. While the delimiters used may be changed, the most commonly used GML delimiters use the character string ":" to open a start tag, ":e" to open an end tag and a period to close a tag. Each of these tags are then mapped, using DCF facilities for this purpose, to collections of DCF control words which are called macros. GML therefore has the capability (that SGML also has) of isolating the processing (the macros) from the data itself, which contains only these generic tags.

Symbols are also supported. References to symbols in data are opened with an ampersand and closed with a period.

GML. See “Generic Markup Language”.

hyperlink. A hyperlink is a generic term for document structures which represent connections with locations in the same or other documents.

HyTime. HyTime is the short name for the “International Standard ISO 10744 — Hypermedia/Time-based Structuring Language”. This is an international standard which codifies the interchange of hypertext/hypermedia documents and the semantics represented in those documents. The standard is based on SGML but has the capability of representing connections or links between virtually any kind of data.

IBM ID Doc. IBM ID Doc is a new, SGML-based, application which is the follow-on to the BookMaster markup language. It supports a much more general processing model so that it may be supported by many different products on different platforms. It is also intended to be extendable to support multiple application domains. It is based on the InfoMaster architecture.

InfoMaster. InfoMaster is an architecture which was designed specifically to support IBM ID Doc and to meet IBM's needs in processing other industry standard markup applications. It consists of a number of architectural forms which represent information classes as well as some special semantics which InfoMaster processing systems must process to support the processing of documents which conform to the InfoMaster architecture.

information class. An information class is a set of information that represents a particular organization of data, like an ordered list, a table, a paragraph, etc.. An information class is represented by one or more elements which identify information which are instances of these classes.

information development. Information development is the name for organizations (at least in IBM) which create the documentation and other product information used by the customer.

ISIL. This is an abbreviation for the name of the internal publishing support that evolved into the IBM product known as BookMaster.

locator. In InfoMaster and IBM ID Doc, a generic term for any indicator of where a particular piece of information is located within a document. Typical locators are page numbers and panel identifiers. Locators are normally generated automatically by the processing

system and may be artifacts of the rendering process, as in the case of page numbers.

markup application. See “application”.

PI. See “processing instruction”.

processing instruction. Processing instructions are data included in an SGML document which are marked as being system-specific commands to the processing system. They do not return data to be included in the SGML document. This is what distinguishes PIs from SDATA. Processing instructions may be included inline in the document data or declared as entities.

prolog. The prolog of an SGML document is where the SGML DTD or document type declaration is found. All elements, attributes, entities, notations and other markup definitions are found in the DTD.

SDATA. SDATA is an SGML concept. It is data that may be included in an SGML document that is marked as system dependent. The SGML model assumes that SDATA content may be processed and that this processing will return data to be included in the document. This returned data cannot contain markup. This characteristic of returning data to be included in the document is what distinguishes SDATA form processing instructions. SDATA may only be defined in entity declarations.

SGML. See “Standard Generalized Markup Language”.

Standard Generalized Markup Language. SGML is an international standard for document markup. It is descended from IBM's GML markup language. Its key characteristics is the segregation of processing information for data and its precise definition of valid data organizations using a meta-markup language called a DTD. For more information see the references given in the front matter of this document.

Like GML, SGML has tags and attributes. In SGML, tags mark the start and end of elements (a concept that GML does not formally support). In addition, SGML supports the referencing of symbols and external data using the concept of entities.

symbol. In GML, a symbol is used to reuse, by reference, a string of data anywhere in the document. A DCF control word is used to define the symbol and then a symbol reference is used to indicate where in the document data the symbol's data should be placed.

tag. In SGML, a tag is used to mark the start or end of an element. In GML, a tag is a marker in the document data which is mapped to a set of DCF instructions, a macro.