CDIF TECHNICAL COMMITTEE

CDIF – XML-based Transfer Format

EIA-PN-XXX
CDIF-DRAFT-XML-V02

EDITOR: JOHANNES ERNST
JUNE 1998

This document represents ongoing work of the CDIF Technical Committee. It is not a proposed or approved standard. Do not specify or claim conformance to this document or standard.

ELECTRONIC INDUSTRIES ASSOCIATION
ENGINEERING DEPARTMENT

NOTICE

EIA Engineering standards and publications are designed to serve the public interest through eliminating misunderstandings between manufacturers and purchasers, facilitating interchangeability and improvement of products, and assisting the purchaser in selecting and obtaining with minimum delay the proper product for his particular need. Existence of such standards and publications shall not in any respect preclude any member or nonmember of EIA from manufacturing or selling products not conforming to such standards and publications, nor shall the existence of such standards and publications preclude their voluntary use by those other than EIA members, whether the standard is to be used either domestically or internationally.

Recommended standards and publications are adopted by EIA without regard to whether or not their adoption may involve patents on articles, materials or processes. By such action, EIA does not assume any liability to any patent owner, nor does it assume any obligation whatever to parties adopting the standard or publication.

EIA INTERIM STANDARDS

EIA Interim Standards contain information deemed to be of technical value to the industry, and are published at the request of the originating Committee without necessarily following the rigorous public review and resolution of comments which is a procedural part of the development of an EIA Recommended Standard.

EIA Interim Standards should be reviewed on an annual basis by the formulating Committee and a decision made on whether to proceed to develop an EIA Recommended Standard on this subject. EIA Interim Standards must be canceled by the Committee and removed from the EIA Standards Catalog before the end of their third year of existence.

This EIA standard is considered to have international standardization implication.

COPYRIGHT 1998

Published by
ELECTRONIC INDUSTRIES ASSOCIATION
Engineering Department
2500 Wilson Blvd.
Arlington, VA 22201

PRICE: Please refer to the current
Catalog of EIA & JEDEC STANDARDS & ENGINEERING PUBLICATIONS
or contact EIA Electronic Information Group
Phone: (703) 907-2545, FAX: (703) 907-7501

ISBN xxx

CONTENTS

FIGURES

TABLES

1 Scope and field of application

1.1 Scope

The CDIF Family of Standards is primarily designed to be used as a description of a mechanism for transferring information between CASE tools. It facilitates a successful transfer when the authors of the importing and exporting tools have nothing in common except an agreement to conform to CDIF. The language that is defined for the Transfer Format also has applicability as a general language for Import/ Export from repositories. The CDIF Integrated Meta-model also has applicability as the basis of standard definitions for use in repositories.

The standards that form the complete family of CDIF Standards are documented in *EIA/IS-106 CDIF - CASE Data Interchange Format - Overview*. These standards cover the overall framework, the transfer format and the CDIF Integrated Meta-model.

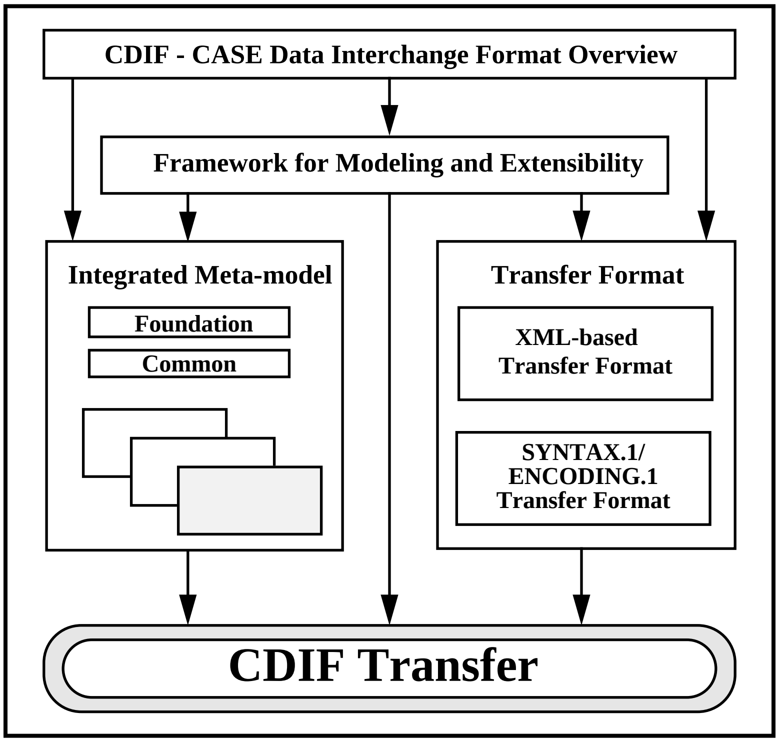

CDIF - CASE Data Interchange Format Overview
Framework for Modeling and Extensibility
Integrated Meta-model
Foundation
Common
Transfer Format
XML-based Transfer Format
SYNTAX.1/ ENCODING.1 Transfer Format
CDIF Transfer

Figure 1
Position in the CDIF Family of Standards

The diagram in Figure 1 depicts the various standards that comprise the CDIF Family of Standards. The shaded box depicts this Standard and its position in the CDIF Family of Standards.

This standard, *EIA/IS-xxx CDIF - XML-based Transfer Format* defines how to exchange CDIF meta-models and CDIF models using the emerging XML file format standard..

1.2 Audience and purpose

This document is intended to be used by anyone wishing to understand and/or use CDIF. This document provides a definition of a single subject area of the CDIF Integrated Meta-model. It is suitable for:

- those evaluating CDIF
- those who wish to understand the principles and concepts of a CDIF transfer
- those developing importers and exporters.

The document *EIA/IS-106 CDIF - CASE Data Interchange Format - Overview* **and the document** *EIA/IS-107 CDIF - Framework for Modeling and Extensibility* **should be read first when initially exploring CDIF and before attempting to read other documents in the CDIF Family of Standards.**

While there are no specific prerequisites for reading this document, it will be helpful for the reader to have familiarity with the following:

- Entity-Relationship-Attribute modeling
- CASE tools
- Information repositories
- Data dictionaries
- Multiple meta-layer modeling.

1.3 Related documents

Editor's Box: insert.
1) CDIF documents
2) XML documents from the W3C

1.4 Structure of this document

This document is organized into the following sections:

Clause 1 'Scope and field of application'

This describes the CDIF Family of Standards and positions the current document within the set of documents.

This describes the intended audience, structure and conventions used in the document.

Clause 2 'References'

The References section contains references to other standards and documents used within this Standard.

Clause 3 'Definitions'

Clause 4 'Notation Conventions'

Clause 5 'Transfer Format Structure'

This describes the structure of an XML-based CDIF Transfer.

Clause 6 'The DTD'

This describes the DTD used for the XML-based CDIF Transfer.

Annex A 'Examples'

This gives one or more examples of how this transfer format is used.

Annex B 'Questions and Answers'

This includes detailed responses to questions that have been raised by those reviewing this document. It is intended that the inclusion of these will aid the reader in understanding this document, and related issues concerning CDIF.

Annex C 'Glossary'

The Glossary contains definitions of terms used within this Standard.

Annex D 'Index'

A comprehensive index is provided.

2 References

Editor's Box: insert EIA and/or ISO standards

3 Definitions

3.1 BNF Conventions

The grammar in this document is defined using an extended version of "Backus Naur Form" (BNF).

In BNF, each syntactic element of the language is defined by means of a "production rule". A "production rule" defines a syntactic element in terms of an expression, ordered left to right, that can be used to form an instance of the element. The expression can consist of characters, character strings, and other syntactic elements.

The version of BNF used in this document has the following conventions:

Table 1: BNF Conventions

Symbol	Meaning
< >	Plain angle braces delimit character strings that are the names of syntactic elemens, i.e., non-terminal symbols. Bold angle braces emboldened are a literal value (i.e., they represents the character string itself).
::=	This is the definition operator. It is used in a production rule to separate the element defined by the rule from its definition. The element being defined is to the left of the operator and the expression that defines the element is to the right of the operator.
[ ]	Square braces enclose optional elements in an expression. The portion of the expression within the braces may be explicitly specified or may be omitted.
{ }	Curly braces group elements in an expression, so that the group can be treated as a single syntactic element.
|	This is the alternative operator. The vertical bar indicates that the portion of the expression to the right of the bar is an alternative to the portion to the left of the bar. If the vertical bar appears at a position where it is not enclosed in curly or square braces, it specifies a complete alternative for the element defined by the production rule. If the vertical bar appears in a portion of the expression enclosed in curly or square braces, it specifies alternatives for the contents of the innermost pair of such braces.
...	The ellipsis indicates that the element to which it applies in a formula may be repeated any number of times, including 0. If the ellipsis appears immediately to the right of a closing curly or square brace, "}" or "]", then it applies to the portion of the expression enclosed between that closing brace and the corresponding opening brace, "{" or "[". If an ellipsis appears immediately to the right of any other element, then it applies only to that element.

Table 1: BNF Conventions

+	The plus sign indicates that the element to which it applies in a formula may be repeated any number of times, but must be repeated at least once.If the ellipsis appears immediately to the right of a closing curly or square brace, "}" or "]", then it applies to the portion of the expression enclosed between that closing brace and the corresponding opening brace, "{" or "[". If an ellipsis appears immediately to the right of any other element, then it applies only to that element.
!!	This is the comment operator. This operator introduces normal English text. It is used as a comment in the BNF and when the definition of a syntactic element is not expressed further in BNF.
<Token>	Anything emboldened and surrounded by plain angle braces is, a token, a BNF non-terminal symbol that is not further defined in the syntax. The detailed representation of a token is specified in the encoding. Terminal Anything emboldened, and not surrounded by angle braces, is a literal value (i.e., it represents the character string itself).
x .. y	This is the range operator. It refers to any symbol in the range x to y. For example, 0..255 refers to all integers 0 to 255 inclusive.

The examples in this document use XML syntax used in the way prescribed by this standard.

Editor's Box: Need to go through and actually do the distinction between bold face and plain angle braces (currently all plain).

---

4 Notation and Conventions

The typographical and naming conventions defined below are used throughout the CDIF Family of Standards.

- Double quotes are used to introduce new terms (e.g., "model layer")
- Bold type is used for emphasis (e.g., this is **important**)
- All meta-objects and meta-meta-objects in CDIF (in meta-models and meta-meta-models) are named by concatenating all the words that name the meta-object or meta-meta-object; the first letter of each word is upper-case, the rest are lower-case. These names are shown in italic (e.g., *MetaAttribute*, *AttributeDerivation*, *IsDrawnUsing*, *IsOptional*)

- The names of objects and values for meta-attributes that appear in examples and instance diagrams are shown in bold italic (e.g., ***Customer***, ***True***, ***100***)

Full details of the CDIF Graphical Notation used in the Meta-model and Meta-meta-model can be found in *EIA/IS-107 CDIF - Framework for Modeling and Extensibility*.

5 Transfer Format Overview

5.1 Description

This section describes, in detail, the exact syntax of each of the major sections of a transfer, together with the syntax of each of the component structures.

5.2 Relationship to XML

The primary application of the XML standards is document markup. Compared to the HTML document type definition, XML-based document type definitions allow for a more strict definition of the content and content structure. On the other hand, the exchange of modeling information according to the CDIF architecture requires an even stricter set of rules to be followed than even in a strict application of XML.

This has two consequences:

- Any CDIF Transfer file employing the XML-based CDIF Transfer Format is also a legal XML file
- The rules to be followed by files employing the XML-based CDIF Transfer Format cannot be fully captured in an XML DTD alone; instead, this standard defines rules that are mandatory for such an XML-based CDIF Transfer beyond those defined in the DTD.

As an example for the latter, consider the rich set of data types for meta-attributes and meta-meta-attributes required to exchange modeling information, and compare them to the significantly smaller set of data types provided by XML. When exchanging models with the XML-based CDIF Transfer Format that employ any of the richer data types, additional rules have to be obeyed.

5.3 Syntax Identifier

The syntax defined in this version of this standard shall have a CDIF Standardized Syntax Identifier of "XML.1", and a Version of "01.00.00". This is used to define the standardized syntax used in a CDIF Transfer (see *EIA/IS-108 CDIF - Transfer Format - General Rules for Syntaxes and Encodings*).

The syntax of the CDIF Syntax ID and the Syntax Version is as follows:

```
<SyntaxID> ::=
    "XML.1"

<SyntaxVersion> ::=
    "01.00.00"
```

5.4 CDIF Transfer Components

5.4.1 Introduction

The general syntax of a CDIF Transfer is as follows:

```
<CDIFTransfer> ::=
    <TransferEnvelope>
    <TransferContents>
```

Figure 2 illustrates the complete structure of a CDIF transfer. A transfer is typically contained in a file, but other means of transfer such as pipes, mailboxes, shared memory, or any other mechanism that can be interpreted as a unicode stream may be used.

5.4.2 The Transfer Envelope

The Transfer Envelope consists of the CDIF Signature, the Syntax Identifier and the Encoding Identifier. As the Transfer Envelope syntax is the same for any CDIF transfer, it is defined in *EIA/IS-108 CDIF - Transfer Format - General Rules for Syntaxes and Encodings*. The syntax identifier for this transfer format is specified in Section 5.3, Syntax Identifier of this document.

5.4.3 The Transfer Contents

The first level of the grammar of the Transfer Contents is the following:

```
<TransferContents> ::=
    <XmlHeader>
    <CONTENT>
        <HeaderSectionClause>
        MetaModelSectionClause
        [ <ModelSectionClause> ]
    </CONTENT>
```

The further levels are detailed in the next section.

5.5 Header Section

5.5.1 XML header

The XML header section includes the information required in the header of a valid XML file. The syntax of the XML header section clause is:

```
<XmlHeader> ::=
    <?XML version=”1.0” RMD=”INTERNAL”>
    <!DOCTYPE CONTENT PUBLIC “http://www.cdif.org/xml/01.00.00.dtd”>
```

Editor’s Box: The draft DTD is available on the CDIF website at http://www.cdif.org/xml/01.00.00-draft3.dtd

---

5.5.2 Introduction

The Header Section defines information that applies to the whole transfer. The syntax of the Header Section clause is:

```
<HeaderSectionClause> ::=
    <HEADER>
        <SummaryClause>
    </HEADER>
```

that is, an introductory keyword followed by a Summary clause, all enclosed within scope delimiters.

5.5.3 Summary Clause

The Summary clause specifies summary information about the transfer, in the form of a number of items. Each item is introduced by an identifier. This standard defines meanings for certain summary section identifiers. Other identifiers may be used, to introduce other items of summary information, but their meanings are not defined in this standard.

The syntax of the Summary clause is:

```
<SummaryClause> ::=
    <SUMMARY>
        [ <IdentifierValuePair> ] ...
    </SUMMARY

<IdentifierValuePair> ::=
    <SUMMARYIDENTIFIER NAME=<SummaryIdentifier>>
        <StringValue>
    </SUMMARYIDENTIFIER>
```

The identifiers must conform to the rules for data type Identifier.

The defined summary section identifiers and their definitions are:

- ExporterName - The name of the exporting software product
- ExporterVersion - The version of the exporting software product
- ExportDate - The date of commencement of the export, in the form YYYY/MM/DD
- ExportTime - The time of commencement of the export, in the form HH:MM:SS
- PublisherName - The name of the person (or function) who initiated the export

An example of a Summary clause is:

```
<SUMMARY>
     <SUMMARYIDENTIFIER Name=”ExporterName”>AutoPal</SUMMARYIDENTIFIER>
     <SUMMARYIDENTIFIER Name=”ExporterVersion”>2.3</SUMMARYIDENTIFIER>
     <SUMMARYIDENTIFIER Name=”ExportDate”>1991/01/12</SUMMARYIDENTIFIER>
     <SUMMARYIDENTIFIER Name=”ExportTime”>13:00:00</SUMMARYIDENTIFIER>
     <SUMMARYIDENTIFIER Name=”PublisherName”>Andrew Shilling</
SUMMARYIDENTIFIER>
     <SUMMARYIDENTIFIER Name=”HardwarePlatform”>Apple Macintosh</
SUMMARYIDENTIFIER>
     <SUMMARYIDENTIFIER Name=”VendorCompanyName”>UltraCASE</
SUMMARYIDENTIFIER>
</SUMMARY>
```

5.6 Meta-model Section

5.6.1 Introduction

The Meta-model Section of the transfer consists of references to standardized Subject Areas, followed by extensions to the Meta-model. The Meta-model for the transfer, known as the "Working Meta-model", is defined by the set of meta-meta-entity and meta-meta-relationship instances that are used in any of the referenced Subject Areas, plus those added by extensions.

As a conveyor of meta-models, this section contains instances of meta-meta-model concepts. The mapping between these concepts, as defined in *EIA/IS-107 CDIF - Framework for Modeling and Extensibility*, and this syntax is straight forward and therefore clarification is included only where necessary.

The syntax of the Meta-model Section clause is:

```
<MetaModelSectionClause> ::=
    <METAMODEL>
        <CDIFSubjectAreaReferenceClause>...
    [ <MetaModelExtensionClause> ]...
</METAMODEL>
```

5.6.2 CDIF Subject Area Reference Clause

The syntax of the CDIF Subject Area Reference clause is:

```
<CDIFSubjectAreaReferenceClause> ::=
    <SUBJECTAREAREFERENCE
        Name=<SubjectAreaName>
        Version=<SubjectAreaVersionNumber>
        [ HREF=<SubjectAreaDtd> ] />

<SubjectAreaName> ::=
    <MetaObjectName>

<SubjectAreaVersionNumber> ::=
    <String>

<SubjectAreaDtd> ::=
    <String>
```

<SubjectAreaName> must be the value of the Name meta-meta-attribute of the relevant SubjectArea instance. <SubjectAreaVersionNumber> must be the value of the VersionNumber meta-meta-attribute of the relevant SubjectArea instance. The optional <SubjectAreaDtd> is a link to an XML file containing the meta-model for this standardized Subject Area. A reference to at least one CDIF Subject Area must be included in the Meta-model Section.

An example of the CDIF Subject Area Reference clause is:

```
<SUBJECTAREAREFERENCE Name=”Foundation” Version=”01.00”>
```

5.6.3 Meta-model Extension Clause

The syntax of the Meta-model Extension clause is:

```
<MetaModelExtensionClause> ::=
    <MetaMetaEntityInstance>
    | <MetaMetaRelationshipInstance>
    | <EnumeratedMetaAttributeExtension>
```

Note that although the syntax allows Meta-meta-entity Instance, Meta-meta-relationship Instance and Enumerated Meta-attribute Extension clauses to be placed in arbitrary order, no forward references are allowed in a transfer.

5.6.4 Meta-meta-entity Instance

Instances of meta-meta-entities are specified by the name of the meta-meta-entity (e.g., MetaEntity, MetaRelationship, MetaAttribute or SubjectArea), followed by a unique CDIF Meta-identifier for the instance, followed by the names and values of the other meta-meta-attributes for the meta-meta-entity.

The syntax of the Meta-meta-entity Instance clause is:

```
<MetaMetaEntityInstance> ::=
    <MME ID=<CDIFMetaIdentifier> Name=<MetaMetaEntityName>>
        [ <MetaMetaAttributeInstance> ] ...
    </MME>

<MetaMetaEntityName> ::=
    <MetaMetaObjectName>

<CDIFMetaIdentifier> ::=
    <Identifier>
```

<MetaMetaEntityName> must be the name of a meta-meta-entity, as defined in *EIA/IS-107 CDIF - Framework for Modeling and Extensibility*.

CDIF Meta-identifiers beginning with a numeric character are reserved for use in the CDIF Integrated Meta-model.

A meta-meta-attribute Instance clause is used to represent each of the meta-meta-attributes (other than the CDIFMetaIdentifier meta-meta-attribute) of the meta-meta-entity.

<MetaMetaAttributeInstance> is defined in the next section.

Several examples of the Meta-meta-entity Instance clause are:

```
(1)  <MME ID=”NEW00001” Name=”SubjectArea”>
         [ <MetaMetaAttributeInstance> ] ...
     </MME>

(2)  <MME ID=”ME001” Name=”MetaEntity”>
         [ <MetaMetaAttributeInstance> ] ...
     </MME>

(3) <MME ID=”MA001” Name=”MetaAttribute”>
         [ <MetaMetaAttributeInstance> ] ...
     </MME>

(4) <MME ID=”MR001” Name=”MetaRelationship”>
         [ <MetaMetaAttributeInstance> ] ...
     </MME>
```

5.6.5 Meta-meta-attribute Instance

Instances of meta-meta-attributes (other than the CDIFMetaIdentifier meta-meta-attribute) are specified by a sequence consisting of the name of the meta-meta-attribute followed by its value.

The syntax of the Meta-meta-attribute Instance clause is:

```
<MetaMetaAttributeInstance> ::=
    <MMA Name=<MetaMetaAttributeName> Value=<MetaMetaAttributeValue>/>
    | <MMA Name=<MetaMetaAttributeName> <MetaMetaAttributeValue> </MMA>

<MetaMetaAttributeName> ::=
    <MetaMetaObjectName>

<MetaMetaAttributeValue> ::=
    <MetaAttributeValue>
```

<MetaMetaAttributeName> must be the name of a meta-meta-attribute associated with the meta-meta-entity. <MetaAttributeValue> is defined in Section ???. The second form of the Meta-meta-attribute Instance clause is employed for meta-meta-attributes of a type that require XML tags, the first form is applied for all others.

Several examples of the Meta-meta-attribute Instance clause are:

```
(1) <MMA Name=”Name” Value=”Function”/>

(2) <MMA Name=”Description”>
      This defines a Business Function which is associated with
      a particular Process or Processes in a Data Flow Model
    </MMA>

(3) <MMA Name=”Aliases” Value=”Business Function,Activity"/>

(4) <MMA Name=”IsOptional” Value=”True”/>
```

5.6.6 Meta-meta-relationship Instance

Instances of meta-meta-relationships are specified by the name of the meta-meta-relationship and the CDIF meta-identifiers for the source and destination meta-meta-entities.

There shall be no forward references to meta-meta-entities defined later in the Meta-model Section. Note that this is a restriction beyond XML.

The syntax of the Meta-meta-relationship Instance clause is:

```
<MetaMetaRelationshipInstance> ::=
    <MMR Name=<FullMetaMetaRelationshipName>
        Src=<SourceMetaMetaEntityCDIFMetaIdentifier>
        Dest=<DestinationMetaMetaEntityCDIFMetaIdentifier>
    [ </MMR> ]

<FullMetaMetaRelationshipName> ::=
    <SourceMetaMetaEntityName>
    .
    <MetaMetaRelationshipName>
    .
    <DestinationMetaMetaEntityName>

<SourceMetaMetaEntityName> ::=
    <MetaMetaObjectName>

<MetaMetaRelationshipName> ::=
    <MetaMetaObjectName>

<DestinationMetaMetaEntityName> ::=
    <MetaMetaObjectName>

<SourceMetaMetaEntityCDIFMetaIdentifier> ::=
    <CDIFMetaIdentifier>

<DestinationMetaMetaEntityCDIFMetaIdentifier> ::=
    <CDIFMetaIdentifier>

<CDIFMetaIdentifier> ::=
    <Identifier>
```

<FullMetaMetaRelationshipName> must be the full name of a meta-meta-relationship, as defined in EIA/IS-107 CDIF - Framework for Modeling and Extensibility.

Several examples of the Meta-meta-relationship Instance clause are:

```
(1) <MMR
      Name=”MetaAttribute.IsLocalMetaAttributeOf.AttributableMetaObject”
      Src=”|MYID002”
      Dest=”|XYZ001”>
    </MMR>

!! This adds a Meta-meta-relationship instance to make the MetaAttribute
   with CDIFMetaIdentifier MYID002 a local meta-attribute of the
   MetaEntity or MetaRelationship with CDIFMetaIdentifier XYZ001.

(2) <MMR
      Name=”CollectableMetaObject.IsUsedIn.SubjectArea”
```

```
            Src=”|MATT006”
            Dest=”|SUBJ001”>
```

5.6.7 Enumerated Meta-attribute Extension

The syntax of the Enumerated Meta-attribute Extension clause is:

```
<EnumeratedMetaAttributeExtension> ::=
    <EXTENDMA Id=<CDIFMetaIdentifier>>
        { <ENUM Value=<EnumeratedIdentifierValue>/> } +
    </EXTENDMA>

<CDIFMetaIdentifier> ::=
    <Identifier>

<EnumeratedIdentifierValue> ::=
    <Identifier>
```

CDIFMetaIdentifier must identify a MetaAttribute with DataType Enumerated defined in the standardized model or earlier in the meta-model extensions. The Enumerated Identifier Values are appended to those values that have already been defined for the MetaAttribute.

An example of the Enumerated Meta-attribute Extension clause is:

```
<EXTENDMA Id=”MA001”>
    <ENUM Value=”Encrypted”/>
    <ENUM Value=”Nonencrypted”/>
</EXTENDMA>
```

5.7 Model Section

5.7.1 Introduction

The Model Section contains references to attributable meta-objects (object types) and actual model data. The object types referenced here are instances of MetaEntities and MetaRelationships that were defined in the Meta-model Section as part of the CDIF Subject Area references or in the Meta-model Extension clauses. The model data are in the form of meta-entity instances, meta-relationship instances and meta-attribute instances.

As a conveyor of models, this section contains instances of meta-model concepts. The mapping between these concepts, as defined in the CDIF Integrated Meta-model, and this syntax is straightforward and therefore clarification is included only where necessary.

All meta-entity and meta-relationship instances in the Model section are uniquely identified by the CDIFIdentifier meta-attribute. The exporter is responsible for ensuring the uniqueness of these identifiers. An importer does not have to retain them, having completed the importation process (successfully or unsuccessfully). CDIFIdentifiers are mapped onto XML IDs.

The syntax of the Model Section clause is:

```
<ModelSectionClause> ::=
    <MODEL>
        <ObjectClause>...
    </MODEL>

<ObjectClause> ::=
    <MetaEntityInstance> | <MetaRelationshipInstance>
```

5.7.2 Meta-entity Instance

Instances of MetaEntities are specified by a sequence consisting of the name of the MetaEntity (as in the meta-model) and a unique identifier for the meta-entity instance, followed by the names and values of any meta-attributes of the meta-entity (as defined in the meta-model).

The syntax of the Meta-entity Instance clause is:

```
<MetaEntityInstance> ::=
    <ME
        {
            Name=<MetaEntityName>
            | ID=<CDIFIdentifier>
        } >
        [ <MetaAttributeInstance> ] ...
    </ME>

<MetaEntityName> ::=
    <MetaObjectName>

<CDIFIdentifier> ::=
    <Identifier>
```

A Meta-entity instance can thus specify its Meta-entity either using the name of the Meta-entity, or its CDIF Identifier. The use of the CDIF Identifier is always possible, the use of the name of the Meta-entity is restricted to those cases where the name of the meta-entity is unique within the Working Meta-model.

An example of the Meta-entity Instance clause is:

```
<ME ID=”MOD01”>
    [ <MetaAttributeInstance> ...
</ME>
```

5.7.3 Meta-relationship Instance

Instances of MetaRelationships are specified by a sequence consisting of the name of the MetaRelationship (as defined in the meta-model), a unique identifier for the instance, a source MetaEntity identifier, a destination MetaEntity identifier, and the names and values of any meta-attributes of the meta-relationship (as defined in the meta-model).

There shall be no forward references to meta-entities defined later in the Model Section.

The syntax of the Meta-relationship Instance clause is:

```
<MetaRelationshipInstance> ::=
    <MR
        {
            Name=<FullMetaRelationshipName>
            | ID=<CDIFIdentifier>
```

```
          }
          Src=<SourceMetaEntityCDIFIdentifier>
          Dest=<DestinationMetaEntityCDIFIdentifier>>
      [ <MetaAttributeInstance> ] ...
      </MR>

  <FullMetaRelationshipName> ::=
      <SourceMetaEntityName>
      .
      <MetaRelationshipName>
      .
      <DestinationMetaEntityName>

  <SourceMetaEntityName> ::=
      <MetaObjectName>

  <MetaRelationshipName> ::=
      <MetaObjectName>

  <DestinationMetaEntityName> ::=
      <MetaObjectName>

  <MetaRelationshipCDIFIdentifier> ::=
      <CDIFIdentifier>

  <SourceMetaEntityCDIFIdentifier> ::=
      <CDIFIdentifierReference>

  <DestinationMetaEntityCDIFIdentifier> ::=
      <CDIFIdentifierReference>

  <CDIFIdentifier> ::=
      <Identifier>

  <CDIFIdentifierReference> ::=
      <IdentifierReference>
```

<SourceMetaEntityName>, <MetaRelationshipName> and <DestinationMetaEntityName> must be the names used in the definition of the meta-relationship.

An example of the Meta-relationship Instance clause is:

```
<MR Name=”DataModel.IsCollectionOf.DataModelObject”
    Src=”|MOD01”
    Dest=”|ENT02”
</MR>
```

5.7.4 Meta-attribute Instance

Instances of MetaAttributes (other than the CDIFIdentifier meta-attribute) are specified by a sequence consisting of the name of the MetaAttribute (as defined in the meta-model), followed by its value.

The syntax of the Meta-attribute Instance clause is:

```
<MetaAttributeInstance> ::=
    <MA Name=<MetaAttributeName> Value=<MetaAttributeValue>/>
    | <MA Name=<MetaAttributeName>><MetaAttributeValue></MA>

<MetaAttributeName> ::=
    <MetaObjectName>
```

The second form of the Meta-meta-attribute Instance clause is employed for meta-attributes of all types that require XML tags, the first form is applied for all others.

An example of the Meta-attribute Instance clause is:

```
<MA Name=”Name” Value=”John Smith”/>
```

5.7.5 Meta-attribute Value

The syntax of the Meta-attribute Value clause is:

```
<MetaAttributeValue> ::=
    <BitmapValue> | <BooleanValue> | <DateValue>
    | <EnumeratedValue> | <FloatValue> | <IdentifierValue>
    | <IntegerValue> | <IntegerListValue> | <PointValue>
    | <PointListValue> | <StringValue> | <TextValue>
    | <TimeValue>
```

5.7.5.1 Bitmap Value

A Bitmap value is defined as a list of pixel values, preceded by the height and width dimensions of the bitmap. A pixel value is a red/green/blue triple representing the intensity of the respective colors. Pixel values in the bitmap are ordered left-to-right and top-to-bottom.

The syntax of the Bitmap Value clause is:

```
<BitmapValue> ::=
    <BITMAP Height=<Height> Width=<Width>>
        <Bitmap>
    </BITMAP>

<Height> ::=
    <PositiveInteger>

<Width> ::=
    <PositiveInteger>

<Bitmap> ::=
   <PixelValue> ...

<PixelValue> ::=
    <PV RED=<PixelRedIntensity>
        GREEN=<PixelGreenIntensity>
        BLUE=<PixelBlueIntensity>/>

<PixelRedIntensity> ::=
    <PixelIntensity>

<PixelGreenIntensity> ::=
    <PixelIntensity>

<PixelBlueIntensity> ::=
    <PixelIntensity>

<PixelIntensity> ::=
    <IntegerValue>
```

The following are examples of valid and invalid bitmap values:

```
(1) <BITMAP Height=2 Width=2>
        <PV RED=”120” GREEN=”50” BLUE=”35”/>
        <PV RED=”130” GREEN=”80” BLUE=”70”/>
        <PV RED=”100” GREEN=”28” BLUE=”231”/>
        <PV RED=”111” GREEN=”255” BLUE=”0”/>
    </BITMAP>
    - Valid

(2) <BITMAP Height=1 Width=2>
        <PV RED=”120” GREEN=”50” BLUE=”35”/>
        <PV RED=”130” GREEN=”80” BLUE=”70”/>
    </BITMAP>
    - Valid
```

5.7.5.2 Boolean Value

The syntax of the Boolean Value clause is:

```
<BooleanValue> ::=
    <TrueValue> | <FalseValue>
```

The following are examples of the Boolean Value clause:

```
(1) "TRUE"

2) "FALSE"
```

5.7.5.3 Date Value

The syntax of the Date Value clause is:

```
<DateValue> ::=
    <DATE Year=<YearValue> Month=<MonthValue> Day=<DayValue>/>

<YearValue> ::=
    <Integer>

<MonthValue> ::=
    <Integer>

<DayValue> ::=
    <Integer>
```

The following are examples of valid and invalid date values:

```
(1) <DATE Year="1940" Month="12" Day="07"/>
    !! Valid
```

5.7.5.4 Enumerated Value

<EnumeratedValue> is represented as a string.

An example of the Enumerated Value clause is:

```
"Red"
```

5.7.5.5 Float Value

A Float value is a decimal number with up to 16 decimal digits of precision within the range of 10^{-1023} to 10^{1023} .

The syntax of the Float Value clause is:

```
<FloatValue> ::=
    <Mantissa> [ <Exponent> ]

<Mantissa> ::=
    [ <Sign> ] <Digit> ...
    [ <DecimalPoint> [ <Digit> ... ] ]

<DecimalPoint> ::=
    .

<Exponent> ::=
    <Exp> <IntegerValue>

<Exp> ::=
    E
```

Several examples of valid Float values are:

```
(1) 2.3

(2) 123.45E2

(3) -123.45E15

(4) +0.23E5

(5) 0.23E-3

(6) 0.23E+3
```

5.7.5.6 Identifier Value

The syntax of the Identifier Value clause is:

```
<IdentifierValue> ::=
    ‟ <StringValjue> ‟
```

An example of the Identifier Value clause is:

```
“johnBrownsBody”
```

5.7.5.7 Identifier Reference Value

The syntax of the Identifier Reference Value clause is:

```
<IdentifierReferenceValue> ::=
    ‟| <StringValue>”
```

This corresponds to the XML XLINK specification with the restriction that only references in the current document are allowed.

An example of the Identifier Reference Value clause is:

```
‟|johnBrownsBody”
```

5.7.5.8 Integer Value

Integer values are expressed as decimal values, binary values, hexadecimal values or octal values.

The syntax of the Integer Value clause is:

```
<IntegerValue> ::=
    [ <Sign> ] <Digit> ...
```

The following are examples of the Integer Values clause:

```
(1) 12345 - positive decimal

(2) -12345 - negative decimal

(3) +12345 - positive decimal

(4) 10101 - positive binary
```

5.7.5.9 Integer List Value

An Integer List is defined as a sequence of integers.

The syntax of the Integer List Value clause is:

```
<IntegerListValue> ::=
    <INTEGERLIST>
        <VAL Value=<IntegerValue>/> +
    </INTEGERLIST>
```

The following are valid and invalid examples of the Integer List Value clause:

```
(1) <INTEGERLIST>
        <VAL Value=”10”>
        <VAL Value=”20”>
        <VAL Value=”11”>
        <VAL Value=”15”>
        <VAL Value=”26”>
        <VAL Value=”32”>
    </INTEGERLIST>
    !! valid

(3) <INTEGERLIST>
        <VAL Value=”10”>
        <VAL Value =”2.4”>
    </INTEGERLIST>
    !! invalid: 2.4 is not an integer value
```

5.7.5.10 Point Value

A Point Value is defined by three integer values representing x, y and z coordinates.

The syntax of the Point Value clause is:

```
<PointValue> ::=
    <POINT XVALUE=<XValue> YVALUE=<YValue> ZVALUE=<ZVALUE>/>

<XValue> ::=
    <Integer>

<YValue> ::=
    <Integer>

<ZValue> ::=
    <Integer>
```

The following are valid and invalid examples of the Point Value clause:

```
(1) <POINT XVALUE=”0” YVALUE=”0” ZVALUE=”0”/>
     !! Valid

(2) <POINT XVALUE=”1” YVALUE=”303” ZVALUE=”292”/>
     !! Valid

(3) <POINT XVALUE=”1” YVALUE=”303”/>
     !! Invalid: all three coordinates must be supplied

(4) <POINT XVALUE=”qq” YVALUE=”eo” ZVALUE=”303”/>
     !! Invalid: must be integer values
```

5.7.5.11 Point List Value

A Point List is defined as a sequence of points.

The syntax of the Point List Value clause is:

```
<PointListValue> ::=
    <POINTLIST>
        <Point> +
    </POINTLIST>
```

The following are examples of the Point List Value clause:

```
(1) <POINTLIST>
        <POINT XVALUE=”0” YVALUE=”0” ZVALUE=”0”/>
        <POINT XVALUE=”1” YVALUE=”1” ZVALUE=”1”/>
        <POINT XVALUE=”2” YVALUE=”2” ZVALUE=”2”/>
    </POINTLIST>
    !! Valid

(2) <POINTLIST>
        <POINT XVALUE=”1” YVALUE=”202” ZVALUE=”292”/>
        <POINT XVALUE=”321” YVALUE=”22” ZVALUE=”33”/>
    </POINTLIST>
    !! Valid
```

5.7.5.12 String Value

A String Value is defined as a character string.

The syntax of the String Value clause is:

```
<StringValue> ::=
    <Char> ...
```

where <Char> is any legal XML character.

An example of the String Value clause is:

```
"This is a string"
```

5.7.5.13 Text Value

The text data type is represented as a group of characters, not necessarily printable nor of any pre-determined maximum length. It is specified as a sequence of one or more text strings; the text value itself is the concatenation of the contents of the individual text strings.

The syntax of the Text Value clause is:

```
<TextValue> ::=
    <Char> ...
```

An example of the Text Value clause is:

```
Program SumIntegers(Input,Output);
var total,input_integer : Integer;
begin
  while not EOF(Input) do
  begin
    ReadLn(input_integer);
    total:=total+input_integer
  end;
WriteLn('Total =',total);
end.
```

This example of the Text Value clause has been laid out with whitespace characters, such as <CR>, transformed for presentation.

5.7.5.14 Time Value

The syntax of the Time Value clause is:

```
<TimeValue> ::=
    <TIME Hour=<Hours> Minutes=<Minutes> Seconds=<Seconds>/>

<Hours> ::=
    <IntegerValue>

<Minutes> ::=
    <IntegerValue>

<Seconds> ::=
    <FloatValue>
```

The following are examples of valid and invalid time values:

```
(1) <TIME Hour=”07” Minutes=”20” Seconds=”23.4”/>
    !! valid

(1) <TIME Hour=”07” Minutes=”20.3” Seconds=”23”/>
    !! invalid: minutes must be integer
```

5.8 Comments

Comments may appear anywhere they are allowed in XML.

6 The DTD

This section describes the formal DTD for the XML-based CDIF Transfer Format. This DTD file will be available electronically from the CDIF Website.

```
<!-- XML-based CDIF Transfer Format DTD -- DRAFT 3 -->
<!-- This document represents ongoing work of the CDIF Technical -->
<!-- Committee.  It is not a proposed or approved standard.  Do  -->
<!-- specify or claim conformance to this document or standard.    -->


<!-- The top-level structure -->
<!ELEMENT Content (Header,  MetaModel, (Model)?)>


<!-- The CDIF header section -->
<!ELEMENT Header (Summary)>

<!-- The Summary Clause -->
<!ELEMENT Summary ((SummaryIdentifier)*)>

<!-- Identifier value pairs for summary -->
<!ELEMENT SummaryIdentifier (#PCDATA)>
<!ATTLIST SummaryIdentifier
    NAME CDATA #IMPLIED>


<!-- The Meta-model section -->

<!ELEMENT MetaModel (SubjectAreaReference, (MME, MMR,
ExtendMetaAttribute)* )>
<!ELEMENT SubjectAreaReference EMPTY>
<!ATTLIST SubjectAreaReference
    Name CDATA #REQUIRED
    Version CDATA #REQUIRED
    HREF IDREF #IMPLIED>

<!-- Meta-meta-entity instance -->
<!ELEMENT MME ((MMA)*)>
<!ATTLIST MME
    ID ID #REQUIRED
    Name CDATA #IMPLIED>

<!-- Meta-meta-relationship instance -->
<!ELEMENT MMR ((MMA)*)>
<!ATTLIST MME
    ID ID #REQUIRED
    Src IDREF #REQUIRED
    Dest IDREF #REQUIRED
    Name CDATA #IMPLIED>

<!-- Meta-meta-attribute instance -->
<!ELEMENT MMA (#PCDATA)>
<!ATTLIST MMA
```

```
    Name CDATA #REQUIRED
    Value CDATA #IMPLIED>

<!-- Extend meta-attribute enumeration -->
<!ELEMENT EXTENDMA ((ENUM)*)>
<!ATTLIST EXTENDMA
    ID ID #REQUIRED>

<!ELEMENT ENUM EMPTY>
<!ATTLIST ENUM
    Value CDATA #REQUIRED>


<!-- Model section -->

<!ELEMENT Model ( ( ME | MR )* )>
<!-- Meta-entity instance -->
<!ELEMENT ME ((MA)*)>
<!ATTLIST ME
    Name CDATA #IMPLIED
    MetaIDRef IDREF #IMPLIED
    ID ID #REQUIRED>

<!ELEMENT MR ((MA)*)>
<!ATTLIST MR
    Name CDATA #IMPLIED
    MetaIDRef IDREF #IMPLIED
    ID ID #REQUIRED
    Src IDREF #REQUIRED
    Dest IDREF #REQUIRED>

<!-- Meta-attribute with value -->
<!ELEMENT MA ( ANY )>
<!ATTLIST MA
    Name CDATA #IMPLIED
    MetaIDRef IDREF #IMPLIED
    Value CDATA #IMPLIED>

<!-- meta-attribute values -->
<!ELEMENT BitMap ((PixelValue)*)>
<!ATTLIST BitMap
    Width CDATA #REQUIRED
    Height CDATA #REQUIRED>

<!ELEMENT PixelValue EMPTY>
<!ATTLIST PixelValue
    Red CDATA #REQUIRED
    Green CDATA #REQUIRED
    Blue CDATA #REQUIRED>

<!ELEMENT DateValue EMPTY>
<!ATTLIST DateValue
    Year CDATA #REQUIRED
    Month CDATA #REQUIRED
    Day CDATA #REQUIRED>
```

```
<!ELEMENT IntegerListValue ((VAL)+)>

<!ELEMENT VAL EMPTY>
<!ATTLIST VAL
    Value CDATA #REQUIRED>

<!ELEMENT PointValue EMPTY>
<!ATTLIST PointValue
    XVALUE CDATA #REQUIRED
    YVALUE CDATA #REQUIRED
    ZVALUE CDATA #IMPLIED>

<!ELEMENT PointList ((PointValue)*)>
<!ELEMENT TimeValue EMPTY>
<!ATTLIST TimeValue
    Hour CDATA #REQUIRED
    Minute CDATA #REQUIRED
    Second CDATA #REQUIRED>
```

Annex G Example

G1. Introduction

Editor's Box: To be done

Annex H QUESTIONS AND ANSWERS

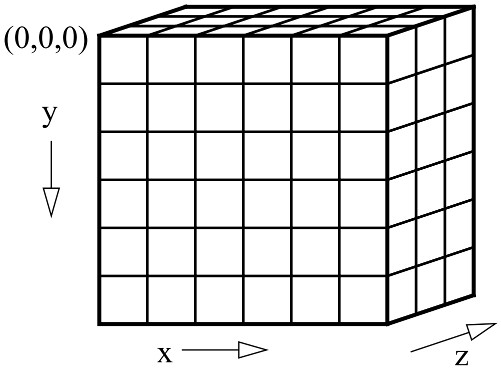
The following questions and answers are presented in order to help the reader's understanding of the standard. They have been derived from discussions held within the EIA/CDIF Division and during presentations to outside observers.

Question 1:

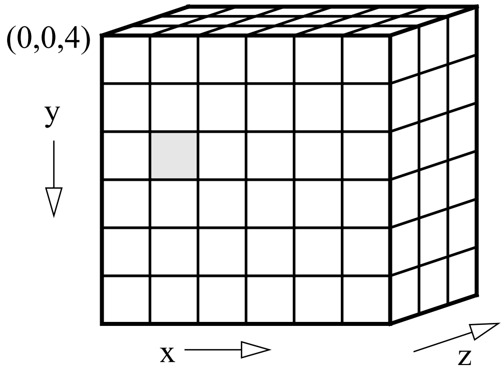
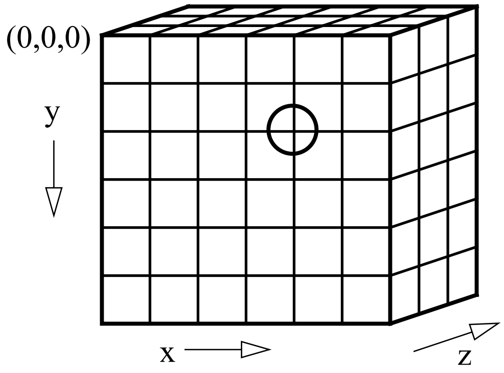
xxx

Answer:

GLOSSARY

ATTRIBUTE	A single-valued characteristic of an entity or relationship.
CDIF	See *CDIF Family of Standards*.
CDIF CLEAR TEXT ENCODING	A human readable encoding (such as ENCODING.1) for a CDIF syntax, such as SYNTAX.1.
CDIF COORDINATE FRAME	The system of graphics addressing used by CDIF. Graphics space is defined as a coordinate system that is a three dimensional rectilinear grid. The intersections of grid lines define addresses (CDIF Points). The intersection of any two grid lines defines a coordinate plane. Each grid cell in a coordinate plane is a CDIF Pixel. All grid coordinates are integers and all grid lines are infinitely thin. The figure below describes the CDIF Coordinate Frame. 
CDIF FAMILY OF STANDARDS	The CDIF Family of Standards is composed of a set of standards that, when used together, provide a standard definition for the interchange of information between CASE tools.
CDIF IDENTIFIER	An attribute that uniquely identifies an object in the Model Section of a transfer.

1.This appendix is not a formal part of the attached EIA Standard but is included for the purposes of information only.

CDIF INTEGRATED META-MODEL	The description of the set of concepts and notations used to define a model. The CDIF Integrated Meta-model defines an Entity-Relationship-Attribute model that is used to construct and define models used in systems development.
CDIF METAIDENTIFIER	A meta-meta-attribute that uniquely identifies a meta-object in the Meta-model Section of a transfer.
CDIF META-META-MODEL	The description of the set of concepts and notations used to define a meta-model. Specifically, the CDIF Meta-meta-model defines an Entity-Relationship-Attribute model that is used to construct and define both meta-models and the CDIF Meta-meta-model itself.
CDIF PIXEL	A grid cell on the coordinate plane in the CDIF Coordinate Frame. The shaded area in the figure below is a CDIF Pixel in the (x,y,4) coordinate plane. 
CDIF POINT	The location of a point in 3-dimensional space as specified by the x,y,z coordinates. The circle in the figure below encloses a point located at (4,2,0). 

CDIF TRANSFER	A combination of a particular syntax, a particular encoding of that syntax, and a meta-model. In other words, a complete definition of the format and contents of a transfer.
CDIF TRANSFER FORMAT	A combination of a particular syntax and a particular encoding of that syntax which together provides a complete definition of the transfer format.
CDIF TRANSFER SYNTAX AND ENCODING	A standard vehicle format supported by CDIF. The combination of SYNTAX.1 and ENCODING.1 forms the initial CDIF Transfer Syntax and Encoding.
CLEAR TEXT	A form of encoding where the resulting physical file is human-readable.
COORDINATE PLANE	See *CDIF Coordinate Frame*.
ENCODING	An encoding defines how the elements of a syntax are physically represented. Details of representation of the various terminal symbols and data types in the syntax's grammar are provided.
ENCODING.1	The CDIF Family of Standards supports multiple transfer formats, each composed of a syntax and an encoding. ENCODING.1 is the primary encoding defined within the CDIF Family of Standards.
ENTITY	An object (i.e., thing, event or concept) that occurs in a model (i.e., transfer).
INFORMATION CONTENT	The Information Content is the set of meta-model and model instances found in a CDIF Transfer.
INSTANCE	An individual instance of a type (e.g., the diagram ***MyDiagram*** is an instance of the *Diagram* type). Usually, an instance is an object which occurs one meta-level beneath its (type's) definition. This is not always true, however (e.g., within the CDIF presentation subject areas where icon type definitions and their instances are both contained within the same meta-level).
INTEGRATED META-MODEL	See *CDIF Integrated Meta-model*.

META-	According to Webster's Ninth New Collegiate Dictionary: *more comprehensive: transcending - used with the name of a discipline to designate a new but related discipline designed to deal critically with the original one.* Meta- is used in CDIF generally as a prefix to a concept to imply definition information about the concept. Specifically, used to designate the location of an object in the three model layers.
META-ATTRIBUTE	A definition of a characteristic of a meta-entity or meta-relationship. Instances of a meta-attribute occur in a model as data values.
META-ENTITY	A definition of a type of data object that occurs in CDIF models. Specifically, a meta-entity represents a set of zero or more meta-attributes, stored together to represent a thing, event or concept that has instances in a model.
META-LAYERS	See *Model Layers*.
META-META-ATTRIBUTE	A definition of a characteristic of a meta-meta-entity or meta-meta-relationship. Instances of a meta-meta-attribute occur in a meta-model as meta-data values.
META-META-ENTITY	A definition of the behavior and structure of meta-entities, meta-relationships, meta-attributes, or subject areas (i.e., a definition of the meta-object definitions used to describe information in models).
META-META-MODEL	See *CDIF Meta-meta-model*.
META-META-RELATIONSHIP	A definition of a type of data object that occurs in CDIF meta-models. Specifically, a meta-meta-relationship represents the definition of a relationship between instances of meta-meta-entities.
META-MODEL	A meta-model contains detailed definitions of the meta-entities, meta-relationships and meta-attributes whose instances represent an actual CDIF transfer.The CDIF Integrated Meta-model (as defined in the set of standards that comprise the CDIF Family of Standards) is a definition of all of the types of information that can be transferred in a CDIF Transfer without using the CDIF extensibility mechanism.
META-OBJECT	A meta-object is a generic term for meta-entities and meta-relationships

META-RELATIONSHIP	A definition of a type of data object that occurs in CDIF models. Specifically, a meta-relationship represents the definition of a relationship between meta-entities that has instances in a model. A meta-relationship may also define a set of zero or more meta-attributes, stored together to represent characteristics of a relationship between meta-entities.
MODEL	A specific collection of software engineering data. This is called a model because it usually represents a model of a software system under development. A collection of instances of meta-objects.
MODEL LAYERS	The different layers of definition (or abstraction) used in defining the CDIF Family of Standards. The four model layers in CDIF are: user data, model, meta-model, meta-meta-model. Any given model layer provides an accurate and complete definition of all the instances that may occur one layer below the given layer. For example, the meta-meta-model provides a set of definitions that are used to construct and understand the meta-model; the meta-model provides a set of definitions that are used to construct and understand a model.
RELATIONSHIP	A real-world association among one or more entities. Where the association is between an entity and itself, the relationship is said to be recursive.
SUBJECT AREA	A related collection of meta-object instance definitions. Subject areas are used to define scoped areas of interest. Subject areas overlap to ensure the integration of the overall Meta-model, but a tool need only use those subject areas relevant to the data to be exported/imported.
TRANSFER	See *CDIF Transfer*.
TRANSFER FORMAT	See *CDIF Transfer Format*.
WORKING META-MODEL	The working meta-model is the definition of the specific meta-objects that may be instantiated in the model section of a CDIF Transfer. The working meta-model comprises the meta-objects in the CDIF Integrated Meta-model that are used by the subject areas referenced in the meta-model section of the transfer, and the meta-objects defined as extensions in the meta-model section.

INDEX