Version 1.1
Metadata Interchange Specification
(MDIS)

August 1, 1997

Version 1.1
Metadata Interchange Specification
(MDIS)

Table of Contents

1.0 Goals and Charter

Goals of the Metadata Interchange Specification Initiative

Situation Analysis
The rapid change in the global economy and an increasingly competitive business climate are driving companies to leverage their information resources in new ways. Enterprise data, once viewed as merely operational or tactical in nature, is now being used for strategic business decision making.

As the rate of business and technological change continues to accelerate, managing this strategic asset and providing timely, accurate, and manageable access to enterprise data becomes increasingly critical. The need to find faster, more comprehensive and efficient ways to access and manage enterprise data has given rise to a variety of new architectures and approaches, including data warehouses, distributed client/server computing, and integrated enterprise-wide applications.

In these environments, **metadata**, or *information about enterprise data*, is emerging as a critical element in effective information resource management. Vendors and users alike recognize the value of metadata, however, the rapid proliferation of data manipulation and management tools has resulted in information technology (IT) products that process metadata differently, and without much consideration for sharing of metadata.

Challenge
To enable full-scale enterprise data management, different IT tools must be able to freely and easily access, update, and share metadata. The only viable mechanism to enable disparate tools from different vendors to exchange metadata is a common metadata interchange specification with guidelines to which the different vendors' tools can comply.

In choosing the interchange-compliant tools, purchasers can be assured of the accurate and efficient exchange of metadata essential to meeting their users' business information needs. This will allow IS managers to build on investments in data management tools and infrastructure with each additional product purchase.

The Metadata Interchange Specification Initiative brings industry vendors and users together to address a variety of problems and issues regarding the exchange, sharing, and management of metadata. This is a voluntary coalition of interested parties with a common focus and shared goals, not a traditional standards body or regulatory group.

Group Charter
To develop a standard for a Metadata Interchange Specification (MDIS) and its support mechanism in such way that it can be implemented within a two- to four-person effort by the average vendor.

This is not intended as a typical standards specification effort, where the goal is to create a standard definition of all the possible information pertinent to the domain and the format for representing it. The assumption here is that for some period of time, at least, the contents of what is considered metadata will be in flux. The most important goal of the MDIS is to define an extensible mechanism that will allow vendors to exchange common metadata as well as carry along "proprietary" metadata.

Group Short-Term Goals

The founding members agreed upon initial goals, including:

• Creating a vendor-independent, industry-defined and -maintained standard access mechanism and standard application programming interface (API) for metadata;

• Enabling users to control and manage the access and manipulation of metadata in their unique environments through the use of interchange specification-compliant tools;

• Allowing users to build tool configurations that meet their needs and to incrementally adjust those configurations as necessary to add or subtract tools without impact on the interchange specification environment;

• Enabling individual tools to satisfy their specific metadata access requirements freely and easily within the context of an interchange model;

• Defining a clean, simple interchange implementation infrastructure that will facilitate compliance and speed adoption by minimizing the amount of modification required to existing tools to achieve and maintain MDIS compliance; and

• Creating a process and procedure not only for establishing and maintaining the MDIS but for extending and updating it over time as required by evolving industry and user needs.

2.0 Organization, Process, and Procedures

Organizational Structure

To achieve the goals of this initiative, vendors and end users are joining forces to drive forward the definition, implementation, and ongoing evolution of an interchange specification. This group of vendors and end users allied with common purpose is known as the Metadata Coalition.

2.1 Membership

The Metadata Coalition
The Metadata Coalition is an open, non-profit organization with functions and processes for business & marketing, and for technical issues surrounding metadata initiatives. Coalition membership is voluntary and open to any company that shares the goals and initiatives of the Coalition and pays the annual dues. The annual dues for membership are $2,500 for software vendor companies and $500 for end-user companies. Dues are the sole source of income to the Coalition and are used to cover Coalition expenses, i.e., meetings, conferences, materials and distribution, and the MDC web page.

Any vendor company whose products create, access, or are dependent on metadata is encouraged to participate in Coalition activities. End users are encouraged to participate to provide the information consumer's perspective, which will help shape a well-rounded and usage-based interchange specification. Coalition member companies may designate any number of participants for discussions, subcommittees, reviews, etc., but each member company represents only one vote.

All extensions and changes to the MDIS, implementation model, or API will be reviewed, discussed, and voted on by Coalition members. The goal of the review and discussion process is to foster consensus by allowing all points of view to be heard and evaluated by Coalition membership. A voting procedure will signify closure of the review and discussion cycle; a simple majority is necessary for a binding vote.

2.2 Organizational Structure

The Metadata Coalition

The Metadata Coalition was established as a not-for-profit corporation. A copy of the Articles of Incorporation and Bylaws are available upon request from the MDC Administrator.

The Metadata Council

Leadership and process administration for the Coalition is provided by the Metadata Council. Council responsibilities include establishing the administrative processes and procedures, setting Coalition goals and objectives, determining deliverables and time frames, setting membership dues, resolving tied votes and deadlocks, creating special-purpose subcommittees as needed, and advising and providing guidance to the general membership, maintaining means for electronic communication between the Coalition membership, and maintaining membership records. The Council will appoint an Administrator who is responsible for managing the overall process.

The Council will consist of a fixed number of companies that represent each of the relevant types or classes of tools and one representative company from the end user community. Any number of people from each Council member company may be designated to participate in meetings, conference calls, "tiger teams," review processes, and so forth. However, there will be only one Council vote recognized per member company.

To preserve the objectivity of the Council, it must always represent the heterogeneity of the marketplace. The objective is to have a mix of vendor members representing a variety of different types of tools. This restriction is important to ensure impartiality of the MDIS and eliminate the risk that Council votes and decisions will be weighted toward the needs or agendas of one type of tool over others. As protection against voting deadlocks, the Council will always comprise an uneven number of members; a binding vote of approval by the Council requires a voting majority.

All of the founding members served on the Council for a term of one year. One of the first tasks addressed by the Council was the election of the end user representative from among a pool of nominations made by Council members. After one year, three of the founding vendor members stepped down while the other three vendor members stayed on for one more year to ensure continuity of process. Council members are now elected annually from among Coalition members to a term of one year for the end user member and two years for the vendor members, promoting a regular rotation of at least half of the vendor members each year.

The Council is co-chaired by two vendor members elected by the Council. Co-chairs serve for a one-year term, unless re-elected.

Subcommittees
Subcommittees are formed as needed for long-term special focus efforts or on a short-term project-specific basis as deemed necessary by the Council. Three ongoing subcommittees have been established: Business/Marketing, Technical, and Finance.

The Business/Marketing Subcommittee is responsible for public relations, communications with the press, the analyst community, and the industry at large, as well as for the preparation and production of marketing and educational materials, progress announcements, group publications, etc. This committee has also created and maintains a Web Page site to facilitate information dissemination, open discussions, and Coalition interaction, including review/comments processes and voting.

The Technical Subcommittee is responsible for the definition, creation, and maintenance of the MDIS language and related technical specifications.

The Finance Subcommittee is staffed by representatives from the accounting and/or finance organizations of two of the Council members. This subcommittee is responsible for developing a high-level budget for the Coalition efforts, approving and reimbursing expenses, keeping the accounting records, and providing biannual financials to the membership.

2.3 Electronic Methods for Communicating

The Metadata Coalition maintains both a Web Page site and an e-mail address to allow members or potential members to communicate electronically. The current Web Page address is:
http://www.he.net/~metadata
which is available through the World Wide Web. The Council also maintains the e-mail address:
coalition@evtech.com
which includes the e-mail addresses of coalition members, and
mdc-spec@evtech.com
for sending comments regarding the MDIS proposal.

Responsibility for the contents of the Web Page is shared jointly by the Business/Marketing Subcommittee and the Technical Subcommittee. The Coalition Administrator is responsible for managing the process.

2.4 Establishing and Maintaining the Specification

The Technical Subcommittee is the keeper ("owner") of the interchange specification (MDIS) for the metadata model and related logical model implementation components and mechanisms.

Any Coalition member may propose an extension or change to the MDIS by sending an e-mail message to:
mdc-spec@evtech.com
These proposals will be reviewed by the Technical Subcommittee and posted, along with the subcommittee's recommendations, on the Web Page. Using these proposals for enhancements as input, the Technical Subcommittee will submit an updated version of the MDIS on an annual basis for review and ratification by the general membership. In this

way, the MDIS can evolve to meet new needs as the types of metadata required by various tools evolve.

The Technical Subcommittee will use the following criteria for determining which proposed extensions should be recommended for inclusion as part of the explicit (i.e., "public") portion of the interchange format:

a. must be generic—vendor and product-independent
b. must be tool-type independent
c. must contribute to data value audit trail/tracking
d. must have relevance across multiple architectures and applications
(e.g., not just DSS warehouse-specific)
e. must be employed by multiple types of tools

The Council is the only body authorized to call for and coordinate Coalition membership votes and is responsible for tallying and reporting results back to the general membership.

The heart of the MDIS is the core set of components that represents the minimum common denominator of metadata elements and the minimum points of integration that must be incorporated into tool products for compliance. Compliance with the MDIS requires support for all relevant core set components and integration points in accordance with the approved specifications.

The MDIS also provides for an approved set of optional/extension components that are relevant only to a particular type or class of tool or a specific application or architecture. Because these are used by more than one tool or application, they can and should conform to the specification definition and set of access parameters, but because they are not generic across all tools, architectures, or applications they would not be eligible for the core set, nor required for compliance.

2.5 Financial Management

Accounting Functions for Metadata Council
The Finance Subcommittee perform the record-keeping and financial functions.

Bank Account
The Finance Subcommittee maintains a bank account at Texas Commerce Bank in the name of Metadata Coalition with signature authority for any one of the following individuals: Linda Hoops, Ken Bartley, Katherine Hammer (all from ETI). This account serves as the operating account for collection of dues and payment of expenses as described in the following procedures. Bank statements are mailed directly to and reconciled by Steve DePasquale (Platinum).

Invoicing/Dues
Applications for new members should be forwarded to the Coalition Administrator and must be accompanied by a check or purchase order for dues amount. Dues covers membership for the calendar year. For dues on mid-year sign-ups, prorata dues will be charged by quarters. For example, new members signing up in the first quarter pay full dues, sign-ups in the second quarter pay 75% of full annual dues, etc. Dues for existing members will be invoiced annually on December 1st to be paid by December 31st.

If dues remain unpaid for 30 days, Council will be notified so that appropriate action can be taken with regard to membership.

Receipts
Checks should be made payable to the Metadata Coalition and mailed to ETI's offices. All checks will be deposited in full into the Metadata Coalition bank account and under no circumstances will cash be received from any part of any deposit. Invoices that have not been paid within 30 days will be reported to the Council for action on membership status.

Disbursements
Invoices or other requests for payment will be submitted to the Coalition Administrator. After approval for payment has been received from the Council, disbursement will be made. All checks will be made payable to the individual or vendor requesting the payment for the approved expenses and under no circumstances will checks be drawn to "Cash."

Budget
The Marketing Subcommittee creates a budget for expenses based on anticipated receipts. Requests for payment of expenses outside the budgeted categories or over budgeted limits will not be paid without Council approval.

Financial Reports/Records
Detailed financial records will be maintained by Linda Hoops in accordance with usual accounting practices.

Financial reports will be prepared by Steve DePasquale on a cash basis and distributed to the Council at least twice annually for the Council's formal meetings.

Annual information or other reports required by the IRS would be prepared by Linda Hoops and/or Steve DePasquale for signature of a Co-chair of the Council.

Tax-exempt Status
The organization has tax exempt status, which affects certain financial and reporting requirements.

3.0 Terminology and Basic Assumptions

3.1 Terminology

The Metadata Interchange Specification draws a distinction between:

- The *Application Metamodel* — the tables, etc., used to "hold" the metadata for schemas, etc., for a particular application; for example, the set of tables used to store metadata in Composer may differ significantly from those used by the Bachman Data Analyst.

- The *Metadata Metamodel*—the set of objects that the MDIS can be used to describe. These represent the information that is common (i.e., represented) by one or more classes of tools, such as data discovery tools, data extraction tools, replication tools, user query tools, database servers, etc. The metadata metamodel should be:
  - Independent of any application metamodel
  - Character-based so as to be hardware/platform-independent

- Fully qualified so that the definition of each object is uniquely identified[1]

3.2 Basic Assumptions

The Metadata Coalition has made the following assumptions:

- Because users' information needs are growing more complex, corporate IS organizations would ideally like the interchange specification to support (to the greatest extent possible) the bidirectional interchange of metadata so that updates can be made in the most natural place. For example, a user might initially specify the source-to-target mapping between a legacy database and a RDBMS target in a CASE tool but, after using a data extraction tool to generate and execute programs to actually move the data, discover that the mapping was somehow incorrect. The most natural place to test out the "fix" to this problem is in the context of the data extraction tool. Once the correction is verified, one updates the *metamodel* in the CASE tool, rather than having to go to the CASE tool, change the mapping, and trigger the metadata interchange between the CASE tool and the data extraction tool before being able to test the new mapping.

- Vendors would like to support the MDIS with a minimum amount of additional development.

In light of these assumptions, the metadata model must be sufficiently extensible to allow a vendor to store the entire metamodel for any application. In other words, MDIS should provide mechanisms for extending the metadata model so that additional (and possibly encrypted) information can be passed. An example of when a vendor might want encryption is in the case of a tool that generates parameters for invoking some internal routine. Because these parameters might provide other vendors with information regarding what is considered a proprietary part of their tool, the vendor may wish to encrypt these parameters.

If one assumed that all updates to the model occurred in the context of a single tool, e.g., the CASE tool in the example above, the MDIS would not benefit from "carrying along" any of the tool-specific metadata. However, as the above example indicates, this assumption is not the "natural" metadata interchange flow. Consequently, some type of mechanism for providing extensions to the type of information exchanged by the interchange specification is necessary if one hopes to achieve bidirectional interchange between vendor applications.

4.0 Metadata Interchange Framework

Overview of Potential Approaches

Implementation of the MDIS metadata model must assume that the metadata itself may be stored in any type of storage facility or format—relational tables, ASCII files, fixed format

[1] Otherwise, something like position in the file would have to determine the "ownership" of certain objects, i.e., that a particular data element identified as "name" is a part of record Y, while another data element identified as "name" is a part of record Z. This requirement will tend to make the standard verbose, but relatively speaking, metadata is not data-intensive; a large company may have hundreds (or thousands) of schemas in use rather than millions.

or customized format repositories, etc. Therefore, the MDIS metadata access methodology must include a framework that will translate an access request into MDIS syntax and format for the metamodel of choice, i.e., the application programming interface's (API) specification parameters.

There are several approaches to consider in accomplishing this:

Procedural Approach

A procedural approach is predicated on each individual tool's interaction with the defined API. It requires that the intelligence to communicate with the API in the specification be built into the tool wherever the tool may need to create, update, access, or otherwise interact with the metadata in the metamodel.

This approach enables the highest degree of flexibility in terms of evolving the standard metadata implementation, as it requires that only the API be modified to accommodate any changes and additions to the MDIS metamodel schema and/or access parameters. However, this approach requires a great deal of up-front effort on the part of the tools vendors to retrofit this logic into the tools to achieve compliance.

Because the tools themselves have to be modified to specifically interact with each given element of the metamodel API, any change in the API must be reflected in an update to the tool. This could put an inordinate and expensive support and maintenance burden on the tools vendors in maintaining compliance as the MDIS inevitably evolves over time.

An example of this approach is the X-Windows user interface standard, which requires all compliant applications to be coded to the X-Windows-specific syntax and argument sequences for calling screen painting and user interaction functions that constitute the X-Windows API. Every time the X-Windows system is changed, every compliant application has to be upgraded to incorporate the new call syntax and arguments in order to maintain compliance.

ASCII Batch Approach

An ASCII Batch approach relies instead on the ASCII file format that contains the description of the common metadata components and standardized access requirements that make up the interchange specification metadata model. In this approach, the entire ASCII file containing the MDIS schema and access parameters is reloaded whenever a tool accesses the metadata through the specification API.

This approach requires only the addition of a simple import/export function to the tools and would not require updating the tool in the event of metadata model changes, because the most up-to-date schema will always be available through the access framework. This eliminates the amount of retrofitting required to enable tools to remain compliant with the MDIS, because the burden for update stays primarily within the framework itself.

However, this approach is resource and process cycle intensive and would likely be prohibitively inefficient, especially in heavy usage scenarios such as decision support data warehouse implementations. It could also have a tangible performance impact and may introduce issues around update coordination in multiple tool access situations. For example, Tool A exports metadata for some external process such as audit reporting; concurrently, Tool B accesses and updates some of those same metadata elements. When Tool A reloads the schema, the updates made by Tool B would be written over and lost.

Hybrid Approach

A hybrid approach that would alleviate these problems to a great degree would follow a data-driven model. By implementing a table-driven API that would support only fully qualified references for each metadata element, a tool could interact with the API through the specification access framework and directly access just the specific metadata object needed. The tables would transparently direct the access path to the required object, so that only that specific object is touched. This also obviates the need for reading in the entire schema. Any changes made would be reflected in the tables, so that tools would not have to be modified to maintain compliance as long as the specification access framework and requirements are not modified.

CDIF Approach
A fourth approach would be to develop the MDIS format in the context of the Electronics Industries Association's CASE Data Interchange Format (CDIF) standard. The CDIF Family of Standards "is primarily a description of a mechanism for transferring information between CASE tools" and supports multiple semantic layers and transfer formats. The current version of the CDIF standards represents a multi-year effort, which expects over time to be adopted as an ISO and ANSI standard. To this end, the goal of the CDIF standard is to be as semantically complete as possible. However, because what constitutes metadata evolves as various types of software technology are developed, the EIA has established an extensible standard and encourages the development of working groups to address new areas of interest. Adopting this approach carries with it two obligations: the Metadata Coalition must appoint one or more members to track the CDIF standards, and every vendor supporting the MDIS format must subscribe to the CDIF publications in order to avoid violating the EIA's copyright on that standard.

Approach recommended
The Metadata Council has recommended the ASCII-based batch approach so that vendors can implement support for the specification with minimum overhead and short time to market. This benefits both vendors and end users by reducing product costs and bringing benefit quickly.

4.1 The Metadata Interchange Specification

There are two basic aspects of the proposed specification:

- Those that pertain to the semantics and syntax used to represent the metadata to be exchanged. These items are those that are typically found in a specifications document.
- Those that pertain to some framework in which the specification will be used. This second set of items is two file-based semaphores that are used by the specification's import and export functions to help the user of the specification control consistency.

Components defining the semantics and syntax that define the specification

The Metamodel
The Metadata Interchange Specification Metamodel describes the entities and relationships that are used to directly represent metadata in the MDIS. The goal in designing this metamodel is twofold:

- To choose the set of entities and relationships that represents the objects that the majority of tools require.

- To provide some mechanism for extensibility in the case that some tool requires the representation of some other type of object. Section 5 describes the metamodel for Version 1.1of the Metadata Interchange Specification. In the rest of this document the entities that are directly represented by the specification are referred to as objects in the "public view," while any other metadata stored in the interchange file is referred to as "private metadata" (i.e., tool-specific metadata).

The mechanism for extending the metamodel
The mechanism chosen to provide extensibility to the specification is analogous to the "properties" object found in LISP environments: a character field of arbitrary length that consists of a set of identifiers and a value, where the identifiers are used by the import function of the specification to locate and identify the private metadata in question and the value is the actual metadata itself. Note: because some tools may consider their private metadata proprietary, the actual value for this metadata may be encrypted.

This approach requires that the import function developed for each tool to support metadata interchange be able to store this proprietary metadata in such a way that for all imported objects that are later exported, the proprietary metadata is associated with any of the imported objects that remain from the original description. This statement assumes that an object may be edited in the context of either tool, resulting in either the creation of new objects or the deletion of imported objects. In the case of new objects, any proprietary metadata required by the original exporting tool will not exist; or, in the case of the deletion of objects, it will no longer be valid. **Note that such an approach requires that the importing product be capable of loading an incomplete or not fully consistent object definition.** This requirement seems reasonable as most tools that support interactive editors must be able to save partially specified entities (say, when the user logs off to go to lunch) and therefore already supports in some capacity the ability to save and retrieve partially specified objects.

The Interchange Specification Access Framework
Version 1.1 of the Metadata Interchange Specification includes information which will support a bidirectional flow of metadata while maintaining metadata consistency. Three types of information are required:

- Versioning information in the header of the file containing the metadata;
- A Tool Profile which describes what type of data elements a tool directly represents and/or updates; and
- A Configuration Profile which describes the "legal flow of metadata." For example, although source-to-target mapping may be specified in the context of some analysis tool, once that metadata has been exported to ETI•EXTRACT and the mapping is changed because of errors found in expected data, one may want to require that all future changes to mapping originate in ETI•EXTRACT. If the configuration profile is set properly, the import function for ETI•EXTRACT would err off if asked to import a conversion specification from the analysis tool with a version number greater than the version number of the one originally imported from the mapping tool.

5.0 The Metadata Interchange Specification Metamodel

Figure 1 illustrates the implied hierarchical structure of the MDIS file. These hierarchies are built embedding each object definition within its parent prior to ending the parent definition. The Figure illustrates the valid objects that can be imbedded within another object. For example, a database can directly have imbedded within it a Dimension, View, Record or Subschema object, but cannot have a direct imbed of an Element or a Level object.

Figure 1:

Hierarchical Model based on "Contains/Contained By" relationships; shows the cardinality of the relationships.

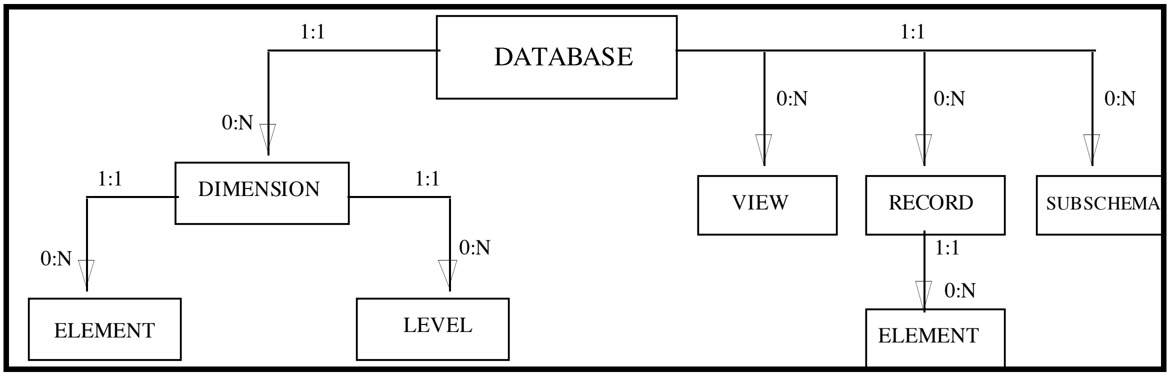

Figure 2 illustrates how to use the relationship object to model certain relationships between objects outside of a hierarchical model. This illustration is not inclusive of all relationship types between objects but is used as the guideline to model the following special circumstances:

To build a logical grouping of record definitions that may map to the same chunk of data. Within the data itself, a key determines what record definition to use. The Subschema is used to group these different record layouts together.

To process the COBOL REDEFINES clause at a structure level, you should use the Redefines relationship type to indicate that one record is redefining another.

To process the COBOL REDEFINES clause at an element level, you should use the Redefines relationship type to indicate that one element is redefining another.

To process COBOL groups use the GroupEquivalent relationship to indicate that a record describes a group element.

Figure 2:

Special Model to handle special cases such as COBOL redefines.

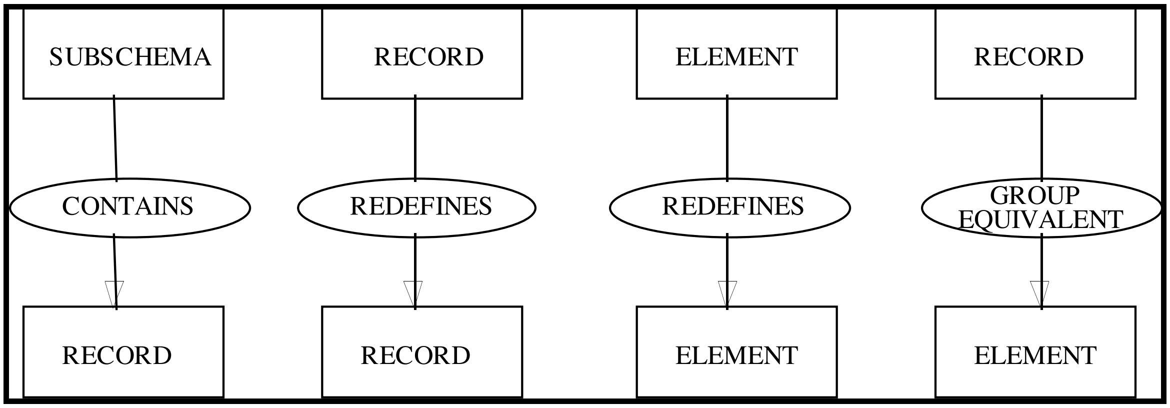

Figure 3 represents the relationships (and cardinality of those relationships) between MDIS object types. Relationships can be between any two supported objects, however, the Meta Data Coalition recommends the following relationship types between objects in addition to those specified in Figure 2. Please note that this is the recommended set of relationships, but other cases may occur where other relationships between objects are necessary._

Figure 3:

OBJECT	RELATIONSHIP TYPE	OBJECT
Database	Equivalent	Database
Subschema	Equivalent	Subschema
Subschema	Equivalent	Record
Subschema	Equivalent	Dimension
Subschema	Equivalent	Element
Record	Equivalent	Record
Record	Equivalent	Dimension
Record	Equivalent	View
Record	Equivalent	Element
Dimension	Equivalent	Dimension
Dimension	Equivalent	Record
Dimension	Equivalent	Element
Element	Equivalent	Element
Element	Equivalent	Record
View	Equivalent	View
View	Equivalent	Record
View	Equivalent	Dimension
Database	Derived	Database
Database	Derived	Subschema
Subschema	Derived	Database
Subschema	Derived	Record
Subschema	Derived	Dimension
Subschema	Derived	View
Record	Derived	Record
Record	Derived	View
Record	Derived	Dimension
Record	Derived	Element
Record	Derived	Subschema
Dimension	Derived	Dimension
Dimension	Derived	Record
Dimension	Derived	Element
Dimension	Derived	View
Dimension	Derived	Subschema
Element	Derived	Element
Element	Derived	Record
Element	Derived	Dimension

OBJECT	RELATIONSHIP TYPE	OBJECT
View	Derived	View
View	Derived	Record
View	Derived	Dimension
Database	Inherits-from	Database
Subschema	Inherits-from	Database
Subschema	Inherits-from	Record
Subschema	Inherits-from	Dimension
Subschema	Inherits-from	View
Record	Inherits-from	Record
Record	Inherits-from	View
Record	Inherits-from	Dimension
Record	Inherits-from	Subschema
Dimension	Inherits-from	Dimension
Dimension	Inherits-from	Record
Dimension	Inherits-from	View
Dimension	Inherits-from	Subschema
Element	Inherits-from	Element
Element	Inherits-from	Record
Element	Inherits-from	View
Element	Inherits-from	Dimension
View	Inherits-from	View
View	Inherits-from	Record
View	Inherits-from	Dimension
Database	Contains	Record
Database	Contains	Subschema
Database	Contains	Dimension
Subschema	Contains	Subschema
Subschema	Contains	Record
Subschema	Contains	Dimension
Subschema	Contains	View
Subschema	Contains	Element
Record	Contains	Record
Record	Contains	Element
Dimension	Contains	Dimension
Dimension	Contains	Element
Element	Contains	Element
View	Contains	View
View	Contains	Record
View	Contains	Dimension
View	Contains	Element
Database	Includes	Record
Database	Includes	Subschema
Database	Includes	Dimension
Subschema	Includes	Subschema
Subschema	Includes	Record

OBJECT	RELATIONSHIP TYPE	OBJECT
Subschema	Includes	Dimension
Subschema	Includes	View
Subschema	Includes	Element
Record	Includes	Record
Record	Includes	Element
Dimension	Includes	Dimension
Dimension	Includes	Element
Element	Includes	Element
View	Includes	View
View	Includes	Record
View	Includes	Dimension
View	Includes	Element
Record	Redefines	Record
Element	Redefines	Element
Record	Group-equivilant	Element
Any Object	User-defined	Any object

6.0 Metadata Interchange Specification (MDIS) MetaObjects

General syntax of interchange statements
MDIS uses a tag language where each MDIS statement begins with a line "BEGIN <statement type>" and ends with a line "END <statement type>." Nesting of different statement types indicates a physical relationship between objects of different types. For example, ELEMENT definitions can be contained within a RECORD definition and RECORD definitions can be contained within a DATABASE definition. Instances of the Relationship Object (see Section 6.7) are used to represent both logical relationships between objects (e.g., a subschema and the record types used by that subschema) and relationships between objects of the same type (e.g., set definitions in a network DBMS or class hierarchies within an OODBMS). For examples of how to represent the different types of objects and relationships used by different data models, see Appendix B.

Please note the following conventions:

- When the angled brackets are used, it means that the text that appears within the brackets should be a value for the type of object referenced. For example, if Arbor were creating the export file and the description of a field value contains <ExportingToolName>, then the value that appears would be ESSBASE.
- When there is a fixed range of values, they are described with the text "VALUE:" followed by the list of legal values separated by commas. For example, VALUE: "PRODUCTION", "DEVELOPMENT".
- The symbols YYYY-MM-DD (using dashes) and HH.MM.SS (using dots) are used to represent the ISO formats for date and time respectively.
- All dates and times should be represented in Greenwich Mean Time (GMT).

- Keywords denoting properties consist of a valid sequence of characters and begin the line on which their value is specified. All keywords must be in English and enumerated values are not translated.
- Many of the values stored in the MDIS file do not have a fixed length. In these cases, the specification declares them of type *varchar*. However, the default maximum text length for a record is 132 bytes, unless specified in the header.
- Long text can be exported from a tool and keep its formatting using the following rules:
  1) Formatting characters:
     a) New lines are specified with the \\n character sequence.
     b) Tabs are specified with the \\t character sequence.
  This is a change from the MDIS 1.0 (which used :CR, :NL, :TAB) to be more in line with current coding specifications.
  2) Long text is enclosed in double quotes and broken into 132 byte records. The record ends with a carriage return/line feed in bytes 131 and 132 if you need to continue the long text across multiple records. A double quote at the end of the text will terminate the long text. Please note that the exporting tool must replace the new lines and tabs in the long text with the \\n and \\t character strings as described in #1 prior.
- Comments can be inserted at any point in an MDIS file by starting each line with "COMMENT". The end of a line ends the COMMENT.
- In listing the object names, a "*" can be used as a wildcard; for example, "NEWTON.DSG..*.*" would mean that the tool could import any metadata associated with the host "NEWTON" and the owner "DSG".
- The BriefDescription field is used to assign descriptive text about the object being defined.

NOTE: Although certain conventions are followed with respect to letter case in this document, key words are not letter case-sensitive.

Granularity of export
Since this interchange mechanism will be executed in batch mode, the following decision was made with respect to the granularity of export. For any object requested in the call to the export function, the function will export all the object instances contained by the object referenced in the function call (i.e., the entire hierarchy under the requested object), as well as all the objects referenced in relationships that are one level across (for the peer relationships).

New Identifier as a means of reducing verbosity
Version 1.1 of MDIS changes the Identifier field from a character string of concatenated fields to a long integer unique within the export file. There are several reasons for this. The previous identifier was not guaranteed to be unique, for example, element names are not necessarily unique within a record. The new integer identifier reduces the verbosity inherent in the old concatenated identifier, which resulted in a string of unbounded length. In addition, the new identifier allows more flexibility in the way that objects relate to each other. For instance, it is now possible to relate two relationships.

6.1 Header

Description: The purpose of the header information in the interchange file is to identify which version of what tool exported the metadata and which version of the MDIS it used in generating the interchange file, as well as the date and time of the export. This information is used by the importing tool, along with the Configuration Profile, to determine whether the import request is legal or should be rejected.

The following describes the fields that constitute the interchange file header:

Name	Value	Required?
CharacterSet	VALUE: "ENGLISH" (for US English), "INTLENG" (for international English, i.e., Canada, UK, Ireland), "GERMAN", "FRENCH","SPANISH" (for Spain and Latin America), "JAPANESE", "SWISS" (for Swiss, German Swiss and French Swiss), "PORTUG" (for Portugal and Brazilian Portuguese), "ITALIAN", "NORDIC" (for Danish, Swedish and Norwegian)	Required
ExportingTool	"<name of tool which created the MDIS file>"	Required
ToolVersion	"<release number of the exporting tool>"	Required
ToolInstanceID	"<integer>"	Required
MDISVersion	"<version number of the MDIS being employed>"	Required
Date	"YYYY-MM-DD"	Required
Time	"HH.MM.SS"	Required
MaxRecLength	"<integer>"	Optional

CharacterSet - Defines the character set used to specify metadata values in the MDIS file.

ExportingTool - A string defining the name of the tool exporting the metadata.

ToolVersion - The release number of the exporting tool.

ToolInstanceID - An identifier used in both the header information and the configuration profile to identify a particular installation of the exporting tool on the server in the case that there are different installations with different privileges, etc.

MDISVersion - Version of MDIS used by the exporting tool.

Date - The date on which the MDIS file was created/written.

Time - The time (in Greenwich Mean Time) at which the MDIS file was created/written.

MaxRecLength - Maximum length of a line in the MDIS file. Default is 132.

Example:

```
BEGIN HEADER
  CharacterSet "FRENCH"
  ExportingTool "IEF Composer"
  ToolVersion "3.1"
 ToolInstanceID "5"
  MDISVersion "1.1"
  Date "1996-03-15"
  Time "14.32.18"
END HEADER
```

6.2 Definition of common properties

Certain properties are common to all object types represented in the MDIS; for example, Identifier, DateCreated, BriefDescription, etc. This section describes the purpose of each of these common properties and the conventions for representing this information.

Identifier
An identifier uniquely identifies an object and is represented as a long integer unique to the MDIS file. This value is required.

DateCreated
The DateCreated property refers to the date that the metadata defining the object was first created in the context of some tool. This value is optional on both import and export. The format for this value is a quoted string of the form "YYYY-MM-DD."

DateUpdated
The DateUpdated property refers to the date that the metadata defining this object was last updated in the context of some tool. This value is optional on both import and export. The format for this value is a quoted string of the form "YYYY-MM-DD."

TimeCreated
The TimeCreated property defines the time at which the metadata for defining the object was created by some tool. This value is optional on both import and export. The format for this value is a quoted string of the form "HH.MM.SS."

TimeUpdated
The TimeUpdated property defines the time at which the metadata for defining the object was last updated by some tool. This value is optional on both import and export. The format for this value is a quoted string of the form "HH.MM.SS."

BriefDescription
The BriefDescription property is used to assign descriptive text about the object being defined. It is used to store the source of the database or file name containing the data definition. The value of this tag can be used in the presentation to end users. This value is

optional for both import and export. The value for this field is a varchar, represented in the MDIS file by the varchar text enclosed in quotation marks.

LongDescription
The LongDescription property is used to assign descriptive text about the object being defined. The value of this tag can be used in the presentation to end users. This value is optional for both import and export. The value for this field is a varchar, represented in the MDIS file by the varchar text enclosed in quotation marks.

ApplicationData
The ApplicationData property allows tools to store an arbitrary amount of tool-specific metadata required for its processing of the object to which it is assigned. This property can be used to associate proprietary (but necessary to some tool) metadata with a particular object; upon agreement/convention between vendors this property can also be used to exchange metadata that falls out of the current MDIS definition. The value for this property is optional for both import and export. The value for each entry in the ApplicationData property is a varchar, represented in the MDIS file by the varchar text enclosed in quotation marks. Each tool's ApplicationData metadata is encapsulated between angle brackets and takes the form:

```
BEGIN ApplicationData
 Tool "tool 1"
 BEGIN ToolAppData

  up to each tool
 END ToolAppData
 Tool "tool 2"
 BEGIN ToolAppData
   kw val
   kw val
 END ToolAppData
END ApplicationData
```

ContactName
The ContactName property refers to the name of a person or department to contact for more information about this object (metadata or data). This property is used to indicate a person or department responsible for the object. It can be used in the presentation to end users. This property value is optional for both import and export. The value for this field is a varchar, represented in the MDIS by the varchar text enclosed in quotation marks.

ServerName
The ServerName property refers to the name of the server or host system where the object resides. This property value is required. The value for this field is a varchar, represented in the MDIS file by the varchar text enclosed in quotation marks.

DatabaseExtendedType
The DatabaseExtendedType property refers to the vendor database name and database version.. This property value is required. The value for this field is a varchar, represented in the MDIS file by the varchar text enclosed in quotation marks.

OwnerName
The OwnerName property refers to the name assigned as the owner of the object being defined. It may contain the user id of an object owner. For example, the owner of the

relational table HRADMIN.EMPLOYEE would be HRADMIN. This property value is required but may be null. The value for this field is a varchar, represented in the MDIS file by the varchar text enclosed in quotation marks.

ObjectTypes

6.3 Database

Description: A database object can be used to represent:

- a group of files
- a relational database
- a network database
- a hierarchical database
- a multi-dimensional database
- an object database

Usage: The database object can contain the record, dimension, view and subschema objects.

The objects can be either physically (e.g., tables within a relational database) or logically (e.g., BDAM files that have customer information stored within them) related together. The Metadata Coalition does not impose physical or logical rules on the tools.

Database

KEYWORD	VALUE	REQUIRED?
Identifier	"<long int>"	Required
ServerName	"<name of server or host on which database resides >"	Required
DatabaseExtendedType	"vendor database name and database version>"	Required
OwnerName	"<owner of the database>"	Required
DateCreated	"YYYY-MM-DD"	Optional
DateUpdated	"YYYY-MM-DD"	Optional
TimeCreated	"HH.MM.SS"	Optional
TimeUpdated	"HH.MM.SS"	Optional
BriefDescription	"<text in quotes>"	Optional
LongDescription	"<text in quotes>"	Optional

KEYWORD	VALUE	REQUIRED?

ApplicationData	BEGIN ApplicationData
	Tool "tool 1"
	BEGIN ToolAppData
	
	up to each tool
	END ToolAppData
	END ApplicationData
		Optional

ContactName	"<name of person to contact>"	Optional
DatabaseName	"<name of database>"	Required
DatabaseLongName	"<text representing the business term for this object>"	Optional
DatabaseStatus	VALUE: "PRODUCTION", "DEVELOPMENT", "TEST"	Optional
DatabaseType	VALUE: "RELATIONAL", "MULTIDIMENSIONAL", "HIERARCHICAL", "FILE", "OBJECT", "NETWORK"	Required

Description of object-specific fields:

DatabaseLongName - The business term used to define this database to end users

This property is used to assign a logical name to the database that is meaningful to end users.

Property value is optional.

Format: The value for this field is a varchar, represented in the MDIS file by the varchar text enclosed in quotation marks.

DatabaseName - Name of local or remote database

This property is used to assign the name of the physical database on the server. It contains the system name of the database.

Property value is required.

Format: The value for this field is a varchar, represented in the MDIS file by the varchar text enclosed in quotation marks.

DatabaseStatus - Current availability status of database

This property is used to assign the status of the database. The status is used by the importing tool to understand the current status of the database and can be used in the presentation to end users.

Property value is optional for both import and export.

Format: "quoted text string". Values:

"PRODUCTION"	database is in production mode
"DEVELOPMENT"	database is in development mode
"TEST"	database is in test mode

DatabaseType - Type of database

This property is used to assign the type of database being defined. The property is used to help the tools understand how the data in the database is physically represented/stored in the source system.

Property value is required.

Format: "quoted text string". Values:

"RELATIONAL"	for a relational database
"MULTIDIMENSIONAL"	for a multidimensional database
"HIERARCHICAL"	for a hierarchical database
"FILE"	for a file based database
"OBJECT"	for a object based database
"NETWORK"	for a network based database

Example:

```
BEGIN DATABASE
   Identifier "001"
   ServerName "NEWTON"
   DatabaseExtendedType "AIX1.0"
   OwnerName "HRADMIN"
   DatabaseName "PAYROLL"
   DateCreated "1992-12-02"
   TimeCreated "23.12.15"
   DateUpdated "1996-03-10"
   TimeUpdated "08.00.00"
   BriefDescription "DB2/MVS payroll database at Newton site"

BEGIN ApplicationData
Tool "DXT"
BEGIN ToolAppData
```

```
CREATE DXT FILENAME=PAYROLL, DESC="DB2/MVS payroll
database at Newton site"ACCESS =GDI,GDIEXIT=GDIDB2S,
GDIXTYPE=SELECT
END ToolAppData
END ApplicationData



  DatabaseStatus "PRODUCTION"
  DatabaseType "RELATIONAL"

    BEGIN RECORD. . . .
END DATABASE
```

6.4 Subschema

Description: The Subschema object is used to provide a logical grouping of record objects that describes a meaningful subset of a database. Instances of the Relationship object (of type "CONTAINS") are used to represent the record types that belong in a particular subschema.

Usage: The Subschema object can be used to represent a logical sub-grouping of components within a database

- logical groupings of relational tables
- logical groupings of files
- logical groupings of objects within an object database
- logical groupings of segments within a hierarchical database
- logical groupings of records within a network database

These objects can represent only logical relationships (e.g., records layouts of a QSAM file that change based upon a key in the data) between objects.

Subschema

KEYWORD	VALUE	REQUIRED?
Identifier	"<long int>"	Required
ServerName	"<name of server or host on which database resides>"	Required
OwnerName	"<group with rights to subschema>"	Required
DatabaseName	"<database name to which the subschema belongs>"	Required
DateCreated	"YYYY-MM-DD"	Optional

KEYWORD	VALUE	REQUIRED?
DateUpdated	"YYYY-MM-DD"	Optional
TimeCreated	"HH.MM.SS"	Optional
TimeUpdated	"HH.MM.SS"	Optional
BriefDescription	"<text>"	Optional
LongDescription	"<text>"	Optional

ApplicationData	BEGIN ApplicationData
	Tool "tool 1"
	BEGIN ToolAppData
	up to each tool
	END ToolAppData
	END ApplicationData
		Optional

ContactName	"<name of person to contact>"	Optional
SubschemaLongName	"<text representing the business term for this metadata>"	Optional
SubschemaName	"<subschema name>"	Required

Description of object-specific fields:

SubschemaLongName - The business term used to define this subschema to end users.

> This property is used to assign a logical name to the subschema which is meaningful to end users.
>
> Property value is optional.
>
> Format: The value for this field is a varchar, represented in the MDIS file by the varchar text enclosed in quotation marks.

SubschemaName - Name of subschema

> This property is used to assign the name for the subschema that is a logical grouping of other objects.
>
> Property value is required.
>
> Format: The value for this field is a varchar, represented in the MDIS file by the varchar text enclosed in quotation marks.

Example:

```
BEGIN SUBSCHEMA
   Identifier "002"
   ServerName "NEWTON"
   OwnerName "PUBLIC"
   DatabaseName "PAYROLL"DateCreated "1992-12-02"
   DateUpdated "1996-03-10"
   TimeUpdated "08.00.00"
   TimeCreated "23.12.15"
   BriefDescription "Grouping of employment data"
    BEGIN ApplicationData
Tool "IEFComposerV1.2"
BEGIN ToolAppData
HOST_TYPEX1243
END ToolAppData
Tool "ESSbaseV3.1"
BEGIN ToolAppData
MAX-NUM-DIM6TIMEOUT30
END ToolAppData
END ApplicationData

   SubschemaName "EMPLOYMENT"
   SubschemaLongName "Employment Info"
  END SUBSCHEMA
```

The usage and representation of subschemas are not inclusive, but simply examples.

6.5 Record

Description: The purpose of the record is to provide a physical grouping of element objects that describe a unit of data.

Usage: The record object is used to represent:

- record layouts of a file (e.g., a COBOL copybook)
- relational database table structures (DDL)
- segment within a hierarchical database
- record within a network database
- object or class definition in an object-oriented database
- group elements in a COBOL file
- multiple layouts of a record (redefines clause)

The record object can contain objects representing:

- columns within a relational table
- properties or objects within an object database
- fields within a record type

Record

KEYWORD	VALUE	REQUIRED?
Identifier	"<long int>"	Required

KEYWORD	VALUE	REQUIRED?
ServerName	"<name of server or host on which database resides>"	Required
OwnerName	"<name of owner>"	Required
DatabaseName	"<name of database>"	Required
TimeCreated	"HH.MM.SS"	Optional
TimeUpdated	"HH.MM.SS"	Optional
BriefDescription	"<text in quotes>"	Optional
LongDescription	"<text in quotes>"	Optional

ApplicationData	BEGIN ApplicationData
	Tool "tool 1"
	BEGIN ToolAppData
	up to each tool
	END ToolAppData
	END ApplicationData
		Optional

ContactName	"<name of person to contact>"	Optional
RecordLongName	"<text representing the business term for this object to end users >"	Optional
RecordName	"<record name>"	Required
RecordLastRefreshDate	"YYYY-MM-DD-HH.MM.SS"	Optional
RecordUpdateFrequency	"<text>"	Optional
RecordType	VALUE: "TABLE", "SEGMENT", "FILE", "CLASS", "RECORD", "GROUP"	Required

Description of object-specific fields:

RecordLongName - The business term used to define this record to end users.

This property is used to assign a logical name to the record that is meaningful to end users.

Property value is optional.

Format: The value for this field is a varchar, represented in the MDIS file by the varchar text enclosed in quotation marks.

RecordName - Name of table, file, segment, or record where data is stored.

This property is used to assign the name of the physical record. This name may be the table, file, segment, or record name. It contains the system name of the record.

Property value is required.

Format: The value for this field is a varchar, represented in the MDIS file by the varchar text enclosed in quotation marks.

RecordLastRefreshDate - The date that the data was last updated.

The value of this keyword is used to identify the date that the data was last updated in the actual database. This value is the date of the last update of the data values for one or more instances of this record type and does not reflect the date that the metadata defining this record was last updated.

Property value is optional for both import and export.

Format: quoted string of format "YYYY-MM-DD-HH.MM.SS".

RecordUpdateFrequency - Frequency of updates to record data

This property is used to identify how frequently the source data is updated. It can be used for presentation to the end users and is a free form text field (e.g., "NIGHTLY", "HOURLY", "WEEKLY", "EVERY FRIDAY", etc.).

Property value is optional for both import and export.

Format: The value for this field is a varchar, represented in the MDIS file by the varchar text enclosed in quotation marks.

RecordType - Type of record

This property is used to identify the type of record being described.

Property value is required.

Format: "quoted text string". Values:

"TABLE"	definition is for a relational table
"SEGMENT"	definition is for a hierarchical segment
"FILE"	definition is for a file record
"CLASS"	definition is for a class in an OODB
"RECORD"	definition is for a record in a network database

"GROUP"   definition for a group item (i.e., a group of contiguous elements within a record which can be referenced/manipulated as a unit)

Example:

```
BEGIN RECORD
  Identifier  "003"
  DateCreated "1992-12-02"
  TimeCreated "23.12.15"
  DateUpdated "1996-03-10"
  TimeUpdated "08.00.00"
  BriefDescription "Employee personal information"

BEGIN ApplicationData
Tool "DXT"
BEGIN ToolAppData
DXTFILE=PAYROLL
END ToolAppData
END ApplicationData
  ServerName "NEWTON"
  OwnerName "HRADMIN"
  DatabaseName "PAYROLL"
  RecordName "EMPLOYEE"
  RecordLongName "Employee Table"
  RecordLastRefreshDate "1996-02-01-12.00.00 "
  RecordType "TABLE"
   BEGIN ELEMENT...
END RECORD
```

6.6 Element

Description: The purpose of the element object is to provide a physical description of the smallest piece of data that should be described. The element represents a data value that is logically or physically represented in the database. Element objects cannot contain any other objects in the object model. They are considered the lowest definable unit of data.

Usage: The element object is used to represent:

- members within a multidimensional database dimension
- columns within a relational table
- attributes or methods in a class
- fields within a file record
- fields within a hierarchical segment or node

Element

KEYWORD	VALUE	REQUIRED?

KEYWORD	VALUE	REQUIRED?
Identifier	"<long int>"	Required
DateCreated	"YYYY-MM-DD"	Optional
DateUpdated	"YYYY-MM-DD"	Optional
TimeCreated	"HH.MM.SS"	Optional
TimeUpdated	"HH.MM.SS"	Optional
BriefDescription	"<text>"	Optional
LongDescription	"<text>"	Optional

ApplicationData	BEGIN ApplicationData
	Tool "tool 1"
	BEGIN ToolAppData
	up to each tool
	END ToolAppData
	END ApplicationData
		Optional

ContactName	"<name of person to contact>"	Optional
ElementLongName	"<text representing the business term for this object to end users>"	Optional
ServerName	"<name of server or host on which database resides>"	Required
DatabaseName	"<database name>"	Required
DimensionName	"<dimension name>"	Required if not a Record
OwnerName	"<name of owner>"	Required
RecordName	"<record name>"	Required if not a Dimension
ElementName	"<element name>"	Required

KEYWORD	VALUE	REQUIRED?
ElementDataType	VALUE: "CHAR", "VARCHAR", "STRING", "TEXT", "BINARY", "SIGNED-INTEGER", “UNSIGNED-INTEGER”, "DECIMAL", "FLOAT", "DATE", "TIME", "TIMESTAMP", "RECORD", "PROGRAM"	Required
ElementPrecision	"<integer>"	Required for decimal
ElementKeyPosition	"<integer, where 0 means not a key>"	Required
ElementLastRefreshDate	"YYYY-MM-DD-HH.MM.SS"	Optional
ElementLength	"<integer representing maximum length of value database[2]>" (includes precision)	Optional
ElementNulls	"T" or "F", representing Boolean	Optional
ElementPosition	"integer representing byte position in record" (zero based)	Optional
ElementOrdinality	VALUE: "1", "N", "<integer>"	Required

Description of object-specific fields:

ElementLongName - The business term used to define this element to end users.

This property is used to assign a logical name to the element that is meaningful to end users.

Property value is optional.

Format: The value for this field is a varchar, represented in the MDIS file by the varchar text enclosed in quotation marks.

ElementName - Name of field, column, or member

This property is used to assign the name of the physical element. This may be the field, column, or member name based upon the type of element. It contains the system name of the element.

[2] The length expressed here should be the length used in the DDL (if any)

Property value is required.

Format: The value for this field is a varchar, represented in the MDIS file by the varchar text enclosed in quotation marks.

DimensionName - Name of dimension

This property is used to assign the name of the dimension in which the element is found.

Property value is required (if not a record).

Format: The value for this field is a varchar, represented in the MDIS file by the varchar text enclosed in quotation marks.

ElementDataType - Datatype of field, column, or member

This property indicates the data type of the element being defined. This is a free format textual field that is populated from the source tool dictionary.

Property value is required.

Format: "quoted text string". Values:

"CHAR"	for fixed character data
"VARCHAR"	for varying character data
"STRING"	for string data
"TEXT"	for text data
"BINARY"	for BLOB data
"SIGNED-INTEGER"	for signed integer data
“UNSIGNED-INTEGER”	for unsigned integer data
"DECIMAL"	for decimal data
"FLOAT"	for floating point data
"DATE"	for date data
"TIME"	for time data
"TIMESTAMP"	for timestamp data
"RECORD"	Element refers to a group element, defined as a Record elsewhere in the MDIS file
"PROGRAM"	for text representing program code
"POINTER"	Properties of a Record of RecordType "CLASS" can refer to the value of one or more object identifiers. These are represented in the Element object by declaring an ElementDataType of "Pointer."

Element Precision - Measure of accuracy.

This property indicates number of digits that occur to the right of decimal.

Property value is required for decimal values (only).

Format: "<integer>"

ElementKeyPosition - If element is used in key, position of element in the key (1-based).

This property indicates whether the element being described is part of a key for the record. If so, it indicates the order position within the key (1-based).

Property value is optional for both import and export.

Format: "<integer>"

ElementLastRefreshDate - The date that some data value in this type of field was last updated

This value of this keyword is used to identify the date that the data was last updated in the actual database. This represents the *data* update date, and does not reflect the date that the *metadata* was last updated.

Property value is optional for both import and export.

Format: quoted string of format "YYYY-MM-DD- HH.MM.SS".

ElementLength - Maximum length of the field, column, or member

This property indicates the fixed or maximum length of the element being defined. This is a numerical field that is populated from the source tool dictionary. The value of this field depends on the datatype of the element being described. The length expressed here should be the length used in the DDL (if any)

Property value is optional.

Format: "quoted numeric string".

ElementNulls - Indicates whether the element can contain null data

This property indicates whether the element being defined can contain null data or not.

Property value is optional for both import and export

Format: "quoted text string" representing Boolean. Values:

"T"	element can have null data
"F"	element cannot have null data

ElementPosition - Position of element within containing object

This property indicates the byte position (zero based) of the element within the containing record or dimension.

Property value is optional for both import and export.

Format: "quoted numeric string".

ElementOrdinality - Number of instances of this element that can occur within a single record instance (e.g., occurs statement in COBOL).

```
Format: "1"           -one instance per record instance.
        "<integer>"   -exactly <integer> instances per record instance.
        "N"           -an arbitrary number of instances per record instance.
```

Property Value is optional (default is 1)

Example:

```
BEGIN ELEMENT
  Identifier "004"
  DateCreated "1992-12-02"
  TimeCreated "23.12.15"
  DateUpdated "1996-03-10"
  TimeUpdated "08.00.00"
  BriefDescription "Employee Identification Number"

        BEGIN ApplicationData
        Tool "DXT"
BEGIN ToolAppData
DXTFILE=PAYROLL
END ToolAppData
END ApplicationData
  ServerName "NEWTON"
  OwnerName "HRADMIN"   DatabaseName "PAYROLL"
  RecordName "EMPLOYEE"
  ElementName "EMPL_ID"
  ElementLongName "Employee Id"
  ElementDataType "SIGNED-INTEGER"
  ElementKeyPosition "000001"
  ElementLastRefreshDate "1996-02-01-12.00.00"
  ElementLength "000002"
  ElementNulls "F"
  ElementOrdinality "1"
END ELEMENT
```

6.7 Relationship

Description: The Relationship object defines a relationship between objects. In many ways, the Relationship object is the most semantically rich and flexible object in the MDIS meta-model. There are nine types of relationships: EQUIVALENT, DERIVED, INHERITS-FROM, CONTAINS, INCLUDES, LINK-TO, REDEFINES, GROUP-EQUIVALENT, and USER-DEFINED. Conventions for using this object to represent the semantics of different data models are illustrated in Appendix B.

Relationship

KEYWORD	VALUE	REQUIRED?
Identifier	"<long int>"	Required
DateCreated	"YYYY-MM-DD"	Optional
DateUpdated	"YYYY-MM-DD"	Optional
TimeCreated	"HH.MM.SS"	Optional
TimeUpdated	"HH.MM.SS"	Optional
BriefDescription	"<text>"	Optional
LongDescription	"<text>"	Optional

ApplicationData	BEGIN ApplicationData
	Tool "tool 1"
	BEGIN ToolAppData
	
	up to each tool
	END ToolAppData
	END ApplicationData
		Optional

ContactName	"<name of person to contact>"	Optional
RelationshipLongName	"<text representing the business term for this object to end users>"	Optional
RelationshipName	"<relationship name>"	Optional
OwnerName	"<name of owner>"	Required
ServerName	"<name of server or host on which database resides>"	Required
SourceObjectIdentifier	"<long int>"	Required
TargetObjectIdentifier	"<long int>"	Required
SourceSequenceOrder	"<integer:integer>" indicating sequence order (if any) out of the maximum number of possible elements in a relationship in the case that more than one source element is used to compute a target value[3]>"	Optional
RelationshipExpression	"<text representing computation>"	Optional

[3] In this case, there will be multiple instances of the Relationship object used.

KEYWORD	VALUE	REQUIRED?
RelationshipType	VALUE: “EQUIVALENT”, "DERIVED", "INHERITS-FROM", "CONTAINS", "INCLUDES", "LINK-TO", "REDEFINES", GROUP-EQUIVALENT", "USER-DEFINED"	Required
RelationshipOrdinality	VALUE: "1:1” ,“1:N", "N:N", "1:<integer>", "<integer>:1"	Required
RelationshipBidirectional	"T" or "F"	Required

Description of object-specific fields:

RelationshipLongName - The business term used to define this relationship to end users.

This property is used to assign a logical name to the relationship that is meaningful to end users.

Property value is optional.

Format: The value for this field is a varchar, represented in the MDIS file by the varchar text enclosed in quotation marks.

RelationshipName -

This property is used to assign the relationship a name separate from its source object type-target object type.

This property is optional.

Format: The value for this field is a varchar, represented in the MDIS file by the varchar text enclosed in quotation marks.

SourceObjectIdentifier - The MDIS identifier that uniquely defines the source data object.

This property is used to identify the source object identifier.

Property value is required.

Format: The value for this field is represented as a long integer.

TargetObjectIdentifier - The MDIS identifier which uniquely defines the target data object.

This property is used to identify the target object identifier.

Property value is required.

Format: The value for this field is represented as a long integer.

SourceSequenceOrder - Integer indicating position of element value amongst the number of source elements used to compute the target value.

Integer indicating sequence order (if any) in the case that more than one source element is used to compute a target value followed by a colon and an integer indicating the total number of source values used to compute the target value.

Property is optional.

Format: "<integer:integer>"

RelationshipExpression - Represents expression (functional logic) used to compute value of target element from values of related source elements.

This property is used to pass the expression used to compute the value of the target element in a relationship of type "DERIVED".

Property value is required if the RelationshipType is "DERIVED".

Format: The value for this field is a varchar, represented in the MDIS file by the varchar text enclosed in quotation marks.

RelationshipType - There are nine types of relationships:

EQUIVALENT, used to indicate that the data values stored in two data elements are equivalent.

DERIVED, used to indicate that the data value stored in the source element has been used to compute the data value stored in the target . (If more than one source value is used, more than one relationship instance is required, and the source sequence position attribute is used to show any ordering that might be required between these source values.)

INHERITS-FROM, used to indicate the relationship between a superclass and a class that inherits attributes from that superclass.

CONTAINS, used to represent ownership between the source object and the target object. Used to represent logical relationships or physical relationships when a target instance has more than one owner.

INCLUDES, used to logically define at what level in hierarchy members can participate in a detail or aggregate.

REDEFINES, used to represent a relationship which maps the same memory address with a different format. (The same physical location is represented two different ways.) For example, COBOL supports the redefinition of record layouts or field definitions. Use the REDEFINES relationship to reflect this type of redefinition in MDIS.

GROUP-EQUIVALENT, used to represent a relationship where an element in a record is actually a group of elements, represented by a record of type "GROUP".

LINK-TO, used to represent a relationship between records in the case that it is not one of ownership.

USER-DEFINED, allows vendor to specify additional relationship types.

Property value is required.

Format: "quoted text string". Values:

“EQUIVALENT”
"DERIVED"
"INHERITS-FROM"
"CONTAINS"
"INCLUDES"
"LINK-TO"
"REDEFINES"
"GROUP-EQUIVALENT"
"USER-DEFINED"

RelationshipOrdinality - Indicates number of target object instances that occur for every instance of a source object.

Property: Property value is required.

Format: "1:1" one instance of target object for each instance of source object
"1:N" many instances of target object for each instance of source object
"1:<integer>" exactly <integer> instances of target object for each instance of source object
"N:N" many target objects for each source object and many source objects for each target object
"<integer>:1" exactly <integer> instances of source object for each instance of target object

RelationshipBidirectional - Indication of whether or not the relationship is bidirectional.

Property value is required.

Format: "T" or "F"

Example of Relationship of RelationshipType “EQUIVALENT”:

Relationship example - EQUIVALENT - an element that is directly generated from a single IMS field.

```
BEGIN RELATIONSHIP
  Identifier "006"
  DateCreated "1992-12-02"
  TimeCreated "23.12.15"
  BriefDescription "DB2/MVS DEPT_BUDGET column from IDMS"

BEGIN ApplicationData
Tool "IDMS"
BEGIN ToolAppData
 Select department_budget from salary_budget
      where division='SWS"and department='FFH'"
END ToolAppData
END ApplicationData
  OwnerName "HRADMIN"
  ServerName "NEWTON"
  SourceObjectIdentifier "0006"
  TargetObjectIdentifier "0007"
  RelationshipExpression "Select department_budget from    salary_budget where
division='SWS' and department='FFH'"
  RelationshipType “EQUIVALENT”
  RelationshipOrdinality "1:1"
  RelationshipBidirectional "T"
END RELATIONSHIP
```

**Example of Relationship of RelationshipType "DERIVED"
Derived columns - an element that is derived from 3 IMS fields.**

```
BEGIN RELATIONSHIP
  Identifier "009"
  DateCreated "1992-12-02"
  TimeCreated "23.12.15"
  BriefDescription "DB2/MVS EMPL_NAME column derived from IMS"

BEGIN ApplicationData
Tool"DXT"
BEGIN ToolAppData
Extract into EMP_NAME
Select EMP_FIRST,EMP_MI,EMP_LAST from IMSPSB2
END ToolAppData
END ApplicationData
  SourceObjectIdentifier "1000"
  TargetObjectIdentifier "2000"
  RelationshipExpression "Select EMP_FIRST,EMP_MI,EMP_LAST"
  RelationshipType "DERIVED"
  RelationshipOrdinality "1:1"
  RelationshipBidirectional "T"
END RELATIONSHIP
```

6.8 Dimension

Description: A dimension is made up of a hierarchy of members, where members are data elements that are referenced by a set of coordinates that uniquely define their position in a hypercube. Each member can belong to more than one hierarchy; in this case the member is said to be shared between hierarchies. Each dimension has one or more levels that can be referenced by name and numbered either from the top of the dimension (in which case it is called the “generation number”) or the bottom of the dimension (in which case it is called the “level number”), or for levels that can be named as containers, "name-level."

The Element object is used to represent members in the dimension and the Relationship object to define the hierarchies to which members belong.

Usage: A dimension is the collection of members that, from the user’s point of view, all have the same type, e.g., sales by store by region by month, expenses by department by month.

Dimension

KEYWORD	VALUE	/REQUIRED?
Identifier	"<long int>"	Required
DateCreated	"YYYY-MM-DD"	Optional
DateUpdated	"YYYY-MM-DD"	Optional
TimeCreated	"HH.MM.SS"	Optional
TimeUpdated	"HH.MM.SS"	Optional
BriefDescription	"<text in quotes>"	Optional
LongDescription	"<text in quotes>"	Optional

ApplicationData	BEGIN ApplicationData
	Tool "tool 1"
	BEGIN ToolAppData
	up to each tool
	END ToolAppData
	END ApplicationData
		Optional

ContactName	"<name of person to contact>"	Optional
DimensionLongName	"<business terms used to identify this object to end users>"	Optional
ServerName	"<name of server or host on which database owning dimension resides>"	Required
DatabaseName	"<database name>"	Required
OwnerName	"<name of owner>"	Required
SubschemaName	"<subschema name>"	Optional

DimensionName	"<dimension name>"	Required
DimensionType	"<name of dimension type>"	Required
DimensionCount	"<integer>"	Required
DimensionLevelCount	"<integer>"	Required

Description of object-specific fields:

DimensionLongName - The business term used to define this dimension to end users.

This property is used to assign a logical name to the dimension that is meaningful to end users.

Property value is optional.

Format: The value for this field is a varchar, represented in the MDIS file by the varchar text enclosed in quotation marks.

DimensionName - Name of dimension

This property is used to assign the name of the dimension definition to the database in question.

Property value is required.

Format: The value for this field is a varchar, represented in the MDIS file by the varchar text enclosed in quotation marks.

DimensionType - The type of the dimension, e.g., currency, account, etc.

This property is used to assign the type of the dimension definition to the dimension.

Property value is required.

Format: The value for this field is a varchar.

DimensionCount - Number of members in the dimension

This property is used to specify the number of members to be found in the dimension. "N" is used to define a potentially unlimited number of members.

Property value is optional.

Format: "<integer>" or "N"

DimensionLevelCount - Number of levels in the dimension

This property is used to specify the number of levels that have been defined for the dimension in question.

Property value is required.

Format: "<integer>"

Example:

```
BEGIN DIMENSION
  Identifier "99999"
  DateCreated "1992-12-02"
  TimeCreated "23.12.15"
  DateUpdated "1996-03-10"
  TimeUpdated "08.00.00"
  BriefDescription "Sales commission by month"
  ServerName "NEWTON"
  OwnerName "HRADMIN"
  DatabaseName "PAYROLL"
  DimensionLongName "Sales Commission"
  DimensionName "COMMISSION"
  DimensionType "CURRENCY"
  DimensionCount "000012"
  DimensionLevelCount "000003"
  BEGIN ELEMENT
        . . .
  BEGIN LEVEL
        . . .
END DIMENSION
```

6.9 Dimension Levels

Description: A dimension is made up of a hierarchy of members, where members are data elements that are referenced by a set of coordinates that uniquely define their position in a hypercube. Each member can belong to more than one hierarchy; in this case the member is said to be shared between hierarchies. Each dimension has one or more levels, that can be referenced by name and numbered from the top of the dimension (in which case it is called the “generation number”) or the bottom of the dimension (in which case it is called the “level number”).

KEYWORD	VALUE
Identifier	"<long int>"
LevelName	"name of level"
LevelNumber	"<integer>"

LevelType	VALUE: "GENERATION NUMBER" (for levels numbered from the top of the dimension), "LEVEL NUMBER" (for levels numbered from the bottom of the dimension), "NAME-LEVEL" (for levels that can be named as containers)

11. Metadata Interchange Specification System Variables

System variables used in the specification provide the capability to assign critical system information. this allows for system-wide configuration information for the implementation of MDIS.

The only current environmental variable is MDIS_PROFILE, which provides the location of the MDIS tool and configuration profiles. The format is shown below:

```
        SET MDIS_PROFILE =filepathname
                where filepathname= the location of the tool and configuration profiles.
```

This variable is set in each machine's configuration file (such as CONFIG.SYS) where the tools are run. The pathname should reference a common file system directory. The directory must contain the tool profile (MDISTOOL.PRO), and the configuration profile (MDISTOOL.CFG).

7.0 Tool Profile

The Tool Profile and the Configuration Profile are two file-based semaphores that, along with the Header information in the interchange file, will allow the Application Programming Interface (API) for the bidirectional MDIS to help the tools maintain metadata consistency. Three types of information need to be represented:

- Versioning information (characterized as date and time of export) in the header of the file containing the metadata.
- The tool profile, which describes what type of data elements a tool directly represents and/or updates.
- The configuration profile, which describes the "legal flow of metadata." For example, although source-to-target mapping may be specified in the context of an analysis tool, once that metadata has been exported to a conversion tool and the mapping is changed because of errors found in expected data, one may want to require that all future changes to mapping originate in the conversion tool. If the configuration profile is set properly, the import function for the conversion tool would err off if asked to import a conversion specification from the analysis tool with a date and time later than those read in the initial import from the analysis tool.

The purpose of the Tool Profile is to allow the tool developer to declaratively specify to what extent the tool supports the objects directly represented by the MDIS (that is, whether or not the tool represents the objects called out in the Specification). The profile information is contained in the filename MDISTOOL.PRO (or CFG, as applicable), and is located in the path as specified in the system variable MDIS_PROFILE. This information includes:

KEYWORD	VALUE	REQUIRED ?

KEYWORD	VALUE	REQUIRED ?
ToolName	"<name of tool>"	Required
ToolVersion	"<version number of tool>"	Required
MDISVersion	"<version number of MDIS supported>"	Required
Database	Boolean: "T" or "F" if represents object	Required
Subschema	Boolean: "T" or "F" if represents object	Required
Dimension	Boolean: "T" or "F" if represents object	Required
Record	Boolean: "T" or "F" if represents object	Required
Element	Boolean: "T" or "F" if represents object	Required
Relationship	Boolean: "T" or "F" if represents object	Required
ExportedApplicationData	Boolena: "T" or "F' if represents object	Required
Level	Boolean: "T" or "F" if represents object	Required
View	Boolean: "T" or "F" if represents object	Required
InvokeImport	"<string representing how to call import function>"(See below.)	Required
InvokeExport	"<string representing how to call export function>" (See below.)	Required

ApplicationData	BEGIN ApplicationData
	Tool "tool 1"
	BEGIN ToolAppData
	
	up to each tool
	END ToolAppData
	END ApplicationData
		Optional

Format for representing function invocation:
The value for the InvokeImport and InvokeExport properties is a string in which %1,%2 and %3 represent the positions into which the three parameters to each function call should be substituted. (See Sec. 9.0 and 10.0.) For example, "mi_import(%1,%2,%3)". The three parameters are:

1. A string containing the name of the identifier of the object being imported/exported.
2. The unique identifier of the object instance being imported/exported (or '*' if there is more than one object represented in the interchange file).
3. The pathname of either 1) the file containing the metadata to be imported, or 2) the pathname of the output file, where pathname refers to the fully

specified name used to access the desired file in the context of this environment.

Example:

```
BEGIN TOOL
  ToolName "DXT"
  ToolVersion "2.5"
  MDISVersion "1.0"
  Database "T"
  Subschema "T"
  Dimension "F"
  Record "T"
  Element "T"
  Relationship "T"
  Level "T"
  View "T"
  InvokeImport  "DIMPORT "%3" TYPE="%1" NAME="%2""
  InvokeExport  "DEXPORT "%3" TYPE="%1" NAME="%2""
END TOOL

BEGIN TOOL
  ToolName "Tool X"
  ToolVersion "7.8"
  MDISVersion "1.0"
  Database "T"
  Subschema "F"
  Dimension "T"
  Record "T"
  Element "T"
  Relationship "F"
  Level "T"
  View "T"
  InvokeImport  "mi_import (\"%1\", \"%2\", \"%3\")"
  InvokeExport  "mi_export (\"%1\", \"%2\", \"%3\")"
END TOOL

BEGIN TOOL
  ToolName "DataGuide"
  ToolVersion "1.1"
  MDISVersion "1.0"
  Database "T"
  Subschema "T"
  Dimension "T"
  Record "T"
  Element "T"
  Relationship "T"
  Level "T"
  View "T"
  InvokeImport  "dguide.exe /IMPORT %3 /OBJTYPE %1 /OBJ %2"
  InvokeEmport  "dguide.exe /EXPORT %3 /OBJTYPE %1 /OBJ %2"
END TOOL
```

8.0 Configuration Profile

The purpose of the configuration profile is to allow the customer to control what types of metadata a particular tool is allowed to import from other tools. Copies of this file can be stored on every host on which a tool that supports MDIS is installed or a single copy can be accessed via a file server. This file is consulted by the IMPORT function of that tool prior to loading metadata from an input file to verify that the user wants the tool in question to import metadata from the tool listed as the exporter of the metadata in the header information of that MDIS file. The profile information is contained in the filename MDISTOOL.PRO (or CFG, as applicable), and is located in the path as specified in the system variable MDIS_PROFILE.

In this way, the user IS organization controls the flow of metadata between the tools they have chosen to integrate. This decision may be based on: 1) the different types of metadata supported by the tools in question, 2) whether one of the tools has to read the source DDL, or 3) simply the customer's chosen methodology in deploying the tools.

KEYWORD	VALUE	REQUIRED ?
TargetToolName	"<name of tool importing metadata>"	Required
	"<version of importing tool>"	Required
	"<particular installation of importing tool>"	Required
SourceToolName	"<name of tool exporting metadata>"	Required
SourceToolVersion	"<version of exporting tool>"	Required
SourceToolInstance	"<particular installation of exporting tool>"	Required
MDISVersion	"<version of MDIS>"	Required
Objects	"<list of meta-object names which can be imported separated by commas>" or "*" if all supported by the MDIS	Required
AllowOverride	"T" or "F", used to indicate whether or not the importing tool can import multiple versions of the same object (i.e., an object with the same identifier) from the same exporting tool	Required

Note: In listing the object names, a "*" can be used as a wildcard; for example, "NEWTON.DSG..*.*" would mean that the tool could import any metadata associated with the host "NEWTON" and the owner "DSG".

Example:

```
BEGIN CONFIGURATION
   TargetToolName "ABC"
   TargetToolVersion "1.0"
    TargetToolInstance "3"
```

```
  SourceToolName "DXT"
  SourceToolVersion "2.5"
  SourceToolInstance "7"
  MDISVersion "1.1"
   Objects "NEWTON.DSG..*.*"
  AllowOverride "T"
END CONFIGURATION

BEGIN CONFIGURATION
  TargetToolName "XYZ"
  TargetToolVersion "1.0"
  TargetToolInstance "2"
  SourceToolName "DataGuide"
  SourceToolVersion "1.1"
  SourceToolInstance "2"
  MDISVersion "1.1"
  Objects "*"
  AllowOverride "T"
END CONFIGURATION

BEGIN CONFIGURATION
  TargetToolName "NBC"
  TargetToolVersion "2.4.1"
   TargetToolInstance "6"
  SourceToolName "Tool X"
  SourceToolVersion "7.8"
  SourceToolInstance "3"
  MDISVersion "1.1"
  Objects "*"
  AllowOverride "T"
END CONFIGURATION
```

9.0 Import Function

The Version 1.1 IMPORT program takes the following three parameters:

- A string containing the name of the object type being imported.
- The unique identifier of the object instance being imported (or ‘*’ if there is more than one object represented in the interchange file).
- The pathname of the file containing the metadata to be imported, where pathname refers to the fully specified name used to access the desired file in the context of this environment.

The IMPORT function should do the following:

1. Tool processes command.
2. Tool checks the header information in conjunction with the configuration profile to determine the following:
    - Whether the importing tool is authorized to import data which has been written by the exporting tool.

- Whether the version of the objects being loaded is later than the one currently stored in the importing tool (if any); if not, return error 212. (See below.)

If either of the above conditions fail, an error message (100 or 213 respectively) is returned and processing stops.

3. Tool optionally processes object type definition.
4. Tool locates object(s) in tag file.
5. Tool maps object(s) to local definition, storing any "private" metadata associated with an object in such a way that it can be reattached to the object upon export (if the object still exists).
6. Tool returns processing code when complete.

- 0 - All OK
- 100 - Illegal metadata source-not allowed to import metadata from tool defined in header information
- 200 - Object type not supported by tool
- 201 - Tag file not found
- 202 - Object(s) specified not found in tag
- 203 - File contains invalid tags
- 204 - Invalid object type definition
- 205 - Invalid object instance
- 206 - Invalid relationship - source object type not found
- 207 - Invalid relationship - target object type not found
- 208 - Invalid relationship - source instance not found
- 209 - Invalid relationship - target instance not found
- 210 - Code page not supported
- 211 - Security level not supported
- 212 - Importing tool does not support this version of the MDIS
- 213 - Time and date of metadata identified in the header is earlier than time & date previously loaded. (A more current version of the data in MDIS exchange file already exists in the context of the importing tool.)
- 1000 - Severe error

Note: The IMPORT function should use the date and time stored in the Header information in the MDIS file to determine whether it has a later "version" of the metadata than that contained in the file and NOT the "LastUpdated" fields associated with various meta-objects, since these fields are optional and may not be updated by exporting tools. Note also that because edits to metadata can take place in the context of various tools, it is possible that some of the private metadata that an importing tool has previously associated with various common metadata objects may now be inconsistent. It is incumbent upon the importing tools to check for such inconsistencies.

Examples:

Example imports from an interchange file on the h: drive and importing all the DATABASE objects in the interchange file that match the specified qualifier.

```
MI_IMPORT.exe (Database, NEWTON.DSG.*, h:\metadata\newton.tag)
DGUIDE.EXE /IMPORT :h\metadata\newton.tag /OBJTYPE Database /OBJ
newton.dsg.*
DIMPORT 'h:\metadata\newton.tag' TYPE='DATABASE'
NAME='newton.dsg.*'
```

```
main {
    Parse input into interchange_file, object_type, object_instance
    Read configuration profile (using environmental variable)

    If object_type or object_instance is not supported
      exit (200)

    Read interchange file header
    If file not found
      exit (201)

    Case:
      Invalid or unknown tags
        exit (203)
      Invalid source tool generated file
        exit (100)
      Invalid or not supported character set
        exit (210)
      Invalid or not supported MDIS version
        exit (212)
    EndCase

    Read private tool metadata dictionary
    If date of source later than interchange file
      exit (213)

    Verify MDIS object definitions
    If definitions do not match
      exit (204)

    Scan file for object_instance(s)
    If object not found
      exit (202)

    Process object_instance(s) requested
    If error processing or parsing
      exit (205)

    Map MDIS tag to private metadata dictionary
    If error mapping
      exit (205)

    Update private metadata dictionary

    /* tool should use ApplicationData that it can parse (its own or    */
    /* another tools) or prompt the user for more information if needed */
Save off interchange ApplicationData (into dictionary or side file)

    If relationship supported

      Scan file for relationships where object_instance is source
     Process relationships
      If target object type not found
        exit (207) - commit if ok, otherwise rollback changes
      If target instance not found
        exit (209) - commit if ok, otherwise rollback changes

    Close interchange file
    exit (0)
   }
```

10. Export Function

The EXPORT function takes the following four parameters:

- A string containing the name of the object type identifier of object being exported.
- The unique identifier of the object instance being exported (or ‘*’ if there is more than one object represented in the interchange file).
- The pathname of the output file.
- ToolInstanceID is the id specified by the user in the configuration profile, which identified the particular installation of a tool that is exporting the metadata.

The EXPORT function should do the following:

1. Tool processes command.
2. Tool writes the header information (i.e., the MDIS system variables) to tagfile, indicating that version A of tool B is writing version C of object D at time E.
3. Tool writes object definition(s) to tagfile.
4. Tool writes object instance(s) to tagfile. During this phase of the API, the tool is responsible for reattaching any "private" metadata associated with any object that it got from importing the object definition.
5. Tool returns processing code when complete.
    - 0 - All OK
    - 1 - Object instance not found
    - 100 - Object type not supported by tool
    - 101 - Unable to write to output tag file
    - 102 - Target tool does not have appropriate security level
    - 1000 - Severe error

Examples:

Example exports of all RECORD objects using the qualifier into an interchange file on the c: drive.

```
MI.EXPORT.exe (Records, NEWTON.DSG..*SEG*.*, c:\records.tag)
DGUIDE.EXE /EXPORT c:\records.tag OBJTYPE Records /OBJ newton.dsg..*SEG*.*
DEXPORT 'c:\records.tag' TYPE='ELEMENT NAME='newton.dsg..*SEG.*'
```

```
main {
    Parse input into interchange_file, object_type, object_instance
    Read configuration profile (using environmental variable)

    If object_type is not supported
      exit (100)

    Read private metadata dictionary for request object_instance(s)
    If object_instance not found
      exit (1)

    Open MDIS file
    If error opening
      exit (101)

    Format and write MDIS header information
    If error writing
      exit (101)

    Format and write MDIS object definitions
    If error writing
      exit (101)

    If this is a previously imported object
      Read stored ApplicationData from dictionary/side file
      Append private data to ApplicationData
      Format and write object_instance(s) as requested
      If error writing
        exit (101)
    Else
      Add private data to ApplicationData
      Format and write object_instance(s) as requested
      If error writing
        exit (101)

    Format and write related (contained) objects to requested object(s)
    If error writing
      exit (101)

    Format and write relationship objects for all Relationship object types
    If error writing
      exit (101)

    Close interchange file
    exit (0)
   }
```

Appendix A
Summary of MDIS Object Definitions

KEYWORD	*VALUE*	*REQUIRED? (YES or NO)*
BEGIN DATABASE		
Identifier	LONG INT	REQUIRED(Y)
DateCreated	DATE	REQUIRED(N)
DateUpdated	DATE	REQUIRED(N)
TimeCreated	TIME	REQUIRED(N)
TimeUpdated	TIME	REQUIRED(N)
BriefDescription	VARCHAR	REQUIRED(N)
LongDescription	VARCHAR	REQUIRED(N)
ApplicationData	VARCHAR	REQUIRED(N)
ContactName	VARCHAR	REQUIRED(N)
ServerName	VARCHAR	REQUIRED(Y)
DatabaseExtendedType	VARCHAR	REQUIRED(Y)
OwnerName	VARCHAR	REQUIRED(Y)
DatabaseName	VARCHAR	REQUIRED(Y)
DatabaseLongName	VARCHAR	REQUIRED(N)
DatabaseStatus	VARCHAR	REQUIRED(N)
DatabaseType	VARCHAR	REQUIRED(Y)
END DATABASE		
BEGIN SUBSCHEMA		
Identifier	LONG INT	REQUIRED(Y)
DateCreated	DATE	REQUIRED(N)
DateUpdated	DATE	REQUIRED(N)
TimeCreated	TIME	REQUIRED(N)
TimeUpdated	TIME	REQUIRED(N)
BriefDescription	VARCHAR	REQUIRED(N)
LongDescription	VARCHAR	REQUIRED(N)
ApplicationData	VARCHAR	REQUIRED(N)
ContactName	VARCHAR	REQUIRED(N)
ServerName	VARCHAR	REQUIRED(Y)
OwnerName	VARCHAR	REQUIRED(Y)
DatabaseName	VARCHAR	REQUIRED(Y)
SubschemaName	VARCHAR	REQUIRED(Y)
SubschemaLongName	VARCHAR	REQUIRED(N)
END SUBSCHEMA		
BEGIN DIMENSION		
Identifier	LONG INT	REQUIRED(Y)
DateCreated	DATE	REQUIRED(N)
DateUpdated	DATE	REQUIRED(N)
TimeCreated	TIME	REQUIRED(N)
TimeUpdated	TIME	REQUIRED(N)
BriefDescription	VARCHAR	REQUIRED(N)

LongDescription	VARCHAR	REQUIRED(N)
ApplicationData	VARCHAR	REQUIRED(N)
ContactName	VARCHAR	REQUIRED(N)
ServerName	VARCHAR	REQUIRED(Y)
OwnerName	VARCHAR	REQUIRED(Y)
DatabaseName	VARCHAR	REQUIRED(Y)
DimensionName	VARCHAR	REQUIRED(Y)
DimensionLongName	VARCHAR	REQUIRED(N)
DimensionType	VARCHAR	REQUIRED(Y)
DimensionCount	INTEGER	REQUIRED(Y)
DimensionLevelCount	INTEGER	REQUIRED(Y)

END DIMENSION

BEGIN RECORD

Identifier	LONG INT	REQUIRED(Y)
DateCreated	DATE	REQUIRED(N)
DateUpdated	DATE	REQUIRED(N)
TimeCreated	TIME	REQUIRED(N)
TimeUpdated	TIME	REQUIRED(N)
BriefDescription	VARCHAR	REQUIRED(N)
LongDescription	VARCHAR	REQUIRED(N)
ApplicationData	VARCHAR	REQUIRED(N)
ContactName	VARCHAR	REQUIRED(N)
ServerName	VARCHAR	REQUIRED(Y)
OwnerName	VARCHAR	REQUIRED(Y)
DatabaseName	VARCHAR	REQUIRED(Y)
RecordName	VARCHAR	REQUIRED(Y)
RecordLongName	VARCHAR	REQUIRED(N)
RecordLastRefreshDate	TIMESTAMP	REQUIRED(N)
RecordUpdateFrequency	VARCHAR	REQUIRED(N)
RecordType	VARCHAR	REQUIRED(Y)

END RECORD

BEGIN ELEMENT

Identifier	LONG INT	REQUIRED(Y)
DateCreated	DATE	REQUIRED(N)
DateUpdated	DATE	REQUIRED(N)
TimeCreated	TIME	REQUIRED(N)
TimeUpdated	TIME	REQUIRED(N)
BriefDescription	VARCHAR	REQUIRED(N)
LongDescription	VARCHAR	REQUIRED(N)
ApplicationData	VARCHAR	REQUIRED(N)
ContactName	VARCHAR	REQUIRED(N)
ServerName	VARCHAR	REQUIRED(Y)
OwnerName	VARCHAR	REQUIRED(Y)
DatabaseName	VARCHAR	REQUIRED(Y)
DimensionName	VARCHAR	REQUIRED(Y) if not RECORD
RecordName	VARCHAR	REQUIRED(Y) if not DIMENSION
ElementName	VARCHAR	REQUIRED(Y)
ElementLongName	VARCHAR	REQUIRED(N)
ElementDataType	VARCHAR	REQUIRED(Y)

ElementKeyPosition	INTEGER	REQUIRED(Y)
ElementLastRefreshDate	TIMESTAMP	REQUIRED(N)
ElementLength	INTEGER	REQUIRED(N)
ElementNulls	VARCHAR	REQUIRED(N)
ElementPosition	INTEGER	REQUIRED(N)
ElementPrecision	INTEGER	REQUIRED (Y-for decimal)
ElementOrdinality	INTEGER	REQUIRED (Y)
END ELEMENT		
BEGIN RELATIONSHIP		
Identifier	LONG INT	REQUIRED(Y)
DateCreated	DATE	REQUIRED(N)
DateUpdated	DATE	REQUIRED(N)
TimeCreated	TIME	REQUIRED(N)
TimeUpdated	TIME	REQUIRED(N)
BriefDescription	VARCHAR	REQUIRED(N)
LongDescription	VARCHAR	REQUIRED(N)
ApplicationData	VARCHAR	REQUIRED(N)
ContactName	VARCHAR	REQUIRED(N)
ServerName	VARCHAR	REQUIRED(Y)
OwnerName	VARCHAR	REQUIRED(Y)
RelationshipName	VARCHAR	REQUIRED(Y)
RelationshipLongName	VARCHAR	REQUIRED(N)
TargetObjectIdentifier	LONG INT	REQUIRED(Y)
RelationshipType	VARCHAR	REQUIRED(Y)
SourceObjectIdentifier	LONG INT	REQUIRED(Y)
SourceSequenceOrder	INTEGER	REQUIRED(N)
RelationshipExpression	VARCHAR	REQUIRED(N)(Y, if derived)
RelationshipOrdinality	INTEGER	REQUIRED(Y)
RelationshipBidirectional	VARCHAR	REQUIRED(Y)
END RELATIONSHIP		

Appendix B

USING MDIS TO REPRESENT DIFFERENT DATA MODELS

B.1 Representing relational databases

In relational systems, all inter-record relationships are achieved via value-based joins, which are expressed by instances of the MDIS Relationship object of RelationshipType “EQUIVALENT.” There are two types of key relationships that one finds in relational schemas:

- Foreign keys, where a primary key from one table is used to associate a particular tuple in that table with one or more related tuples in the other table
- Compound keys, where a tuple can only be uniquely identified by means of a sequence of foreign keys.

The following MDIS example describes both 1) the tables that comprise a database called COURSE_CATALOG, which illustrates the two types of keys described above, and 2) the MDIS representation of this schema

```
COMMENT /Contains id, name and id of chair for each department
COMMENT CREATE TABLE DEPT
COMMENT (
COMMENT DEPT_ID                 VARCHAR NOT NULL,
COMMENT DEPT_NAME               VARCHAR,
COMMENT DEPT_CHAIR              SMALLINT
COMMENT);
COMMENT
COMMENT Contains id, name and description of each course offered by the university,
COMMENT where DEPT_ID and COURSE_NO serve as a compound key
COMMENT CREATE TABLE COURSE
COMMENT (
COMMENT DEPT_ID                         VARCHAR NOT NULL,
COMMENT COURSE_NO                       SMALLINT NOT NULL,
COMMENT COURSE_NAME                     VARCHAR ,
COMMENT COURSE_DESC                     VARCHAR
COMMENT);
COMMENT
COMMENT
COMMENT  Contains a listing of course offerings, where DEPT_ID, COURSE_NO and
COMMENT  SECTION_NO serve as a compound key
COMMENT CREATE TABLE OFFERINGS
COMMENT (
COMMENT DEPT_ID                 VARCHAR NOT NULL,
COMMENT COURSE_NO               SMALLINT NOT NULL,
COMMENT SECTION_NO              SMALLINT NOT NULL,
COMMENT TIME                    VARCHAR,
COMMENT LOCATION                VARCHAR,
COMMENT FACULTY_SSN             VARCHAR
COMMENT);
COMMENT
```

```
COMMENT Contains faculty info including correspondence between the ID COMMENT
found in the IMS database and the faculty member’s SSN

COMMENT CREATE TABLE FACULTY
COMMENT (
COMMENT FACULTY_OLD_ID      VARCHAR  NOT NULL,
COMMENT FACULTY_SSN         SMALLINT NOT NULL,
COMMENT NAME                VARCHAR,
COMMENT ADDRESS             VARCHAR,
COMMENT RANK                VARCHAR
COMMENT);

BEGIN DEFINITION

COMMENT    Definition of MDIS model goes here....

END DEFINITION

BEGIN DATABASE
    Identifier "001"
    DateCreated “1995-04-12”
    TimeCreated “02.00.00”
    DateUpdated “1996-03-10”
    TimeUpdated “08.00.00”
    BriefDescription “DB2 database containing department scheduling & faculty info”
    ServerName “EINSTEIN”
    DatabaseExtendedType "AIX1.0"
    OwnerName “SYSADMIN”
    DatabaseName “COURSE_CATALOG”
    DatabaseStatus “DEVELOPMENT”
    DatabaseType “RELATIONAL”
COMMENT MDIS description of tables
    BEGIN RECORD
            Identifier "002"
            DateUpdated “1996-03-10”
            TimeUpdated “08.00.00”
            BriefDescription  “Record describing University department”
            RecordName “DEPT”
            RecordLongName “General department data”
            RecordLastRefreshDate “1996-02-01-12.00.00.00”
            RecordType “TABLE”
            BEGIN ELEMENT
                Identifier  "003"
                DateUpdated “1996-03-10”
                TimeUpdated “08.00.00”
                BriefDescription “Unique 4-char key identifying department”
                ElementName “DEPT_ID”
                ElementLongName “Department ID”
                ElementDataType “VARCHAR”
                ElementKeyPosition “1”
                ElementLastRefreshDate “1996-02-01-12.00.00”
                ElementLength “4”
                ElementNulls “F”
            END ELEMENT
```

```
        BEGIN ELEMENT
            Identifier  "004"
            DateUpdated “1996-03-10”
            TimeUpdated “08.00.00”
            BriefDescription “Name of department”
            ElementName “DEPT_NAME”
            ElementLongName “Department name”
            ElementDataType “VARCHAR”
            ElementKeyPosition “0”
            ElementLastRefreshDate “1996-02-01-12.00.00.000000”
            ElementLength “20”
            ElementNulls “F”
        END ELEMENT
        BEGIN ELEMENT
            Identifier "005"
            DateUpdated “1996-03-10”
            TimeUpdated “08.00.00”
            BriefDescription “SSN of individual currently serving as department chair”
            ElementName “CHAIR”
            ElementLongName “Chairman ID”
            ElementDataType “INTEGER”
            ElementKeyPosition “0”
            ElementLastRefreshDate “1996-02-01-12.00.00.000000”
            ElementLength “9”
            ElementNulls “T”
        END ELEMENT
END  RECORD
BEGIN RECORD
        Identifier  "006"
        DateUpdated “1996-03-10”
        TimeUpdated “08.00.00”
        BriefDescription  “Record describing university course”
        RecordName “COURSE”
        RecordLongName “General course  data”
        RecordLastRefreshDate “1996-02-01-12.00.00.000000”
        RecordType “TABLE”
        BEGIN ELEMENT
            Identifier  "007"
            DateUpdated “1996-03-10”
            TimeUpdated “08.00.00”
            BriefDescription “Unique 4 char key identifying department”
            ElementName “DEPT_ID”
            ElementLongName “Department ID”
            ElementDataType “VARCHAR”
            ElementKeyPosition “1”
            ElementLastRefreshDate “1996-02-01-12.00.00.000000”
            ElementLength “4”
            ElementNulls “F”
        END ELEMENT
        BEGIN ELEMENT
            Identifier
            "008"
            DateUpdated “1996-03-10”
            TimeUpdated “08.00.00”
            BriefDescription “Integer identifying course”
            ElementName “COURSE_NO”
            ElementLongName “Course Number”
            ElementDataType “INTEGER”
            ElementKeyPosition “2”
            ElementLastRefreshDate “1996-02-01-12.00.00.000000”
            ElementLength “4”
            ElementNulls “F”
        END ELEMENT
        BEGIN ELEMENT
            Identifier  "009"
            DateUpdated “1996-03-10”
            TimeUpdated “08.00.00”
            BriefDescription “Name of course”
            ElementName “COURSE_NAME”
            ElementLongName “Course name”
            ElementDataType “VARCHAR”
            ElementKeyPosition “0”
            ElementLastRefreshDate “1996-02-01-12.00.00.000000”
            ElementLength “20”
            ElementNulls “F”
        END ELEMENT
        BEGIN ELEMENT
            Identifier  "010"
            DateUpdated “1996-03-10”
            TimeUpdated “08.00.00”
            BriefDescription “Description of course”
            ElementName “DESC_NAME”
            ElementLongName “Course Description"
            ElementDataType “VARCHAR”
            ElementKeyPosition “0”
            ElementLastRefreshDate “1996-02-01-12.00.00.000000”
            ElementLength “60”
            ElementNulls “T”
        END ELEMENT
    END  RECORD
    BEGIN RECORD
        Identifier "011"
        DateUpdated “1996-03-10”
        TimeUpdated “08.00.00”
        BriefDescription  “Record describing course offering information”
        RecordName “OFFERINGS”
        RecordLongName “Course offering data”
        RecordLastRefreshDate “1996-02-01-12.00.00.000000”
        RecordType “TABLE”
        BEGIN ELEMENT
            Identifier
            "012"
            DateUpdated “1996-03-10”
            TimeUpdated “08.00.00”
            BriefDescription “Unique 4-char key identifying department”
            ElementName “DEPT_ID”
            ElementLongName “Department ID”
            ElementDataType “VARCHAR”
            ElementKeyPosition “1”
            ElementLastRefreshDate “1996-02-01-12.00.00.000000”
            ElementLength “4”
            ElementNulls “F”
        END ELEMENT
        BEGIN ELEMENT
            Identifier
            "013"
            DateUpdated “1996-03-10”
            TimeUpdated “08.00.00”
            BriefDescription “Unique integer key identifying course”
            ElementName “COURSE_NO”
            ElementLongName “Course Number”
            ElementDataType “INTEGER”
            ElementKeyPosition “2”
            ElementLastRefreshDate “1996-02-01-12.00.00.000000”
            ElementLength “4”
            ElementNulls “F”
            END ELEMENT
        BEGIN ELEMENT
            Identifier  "014"
            DateUpdated “1996-03-10”
            TimeUpdated “08.00.00”
            BriefDescription “Integer identifying OFFERINGS”
            ElementName “SECTION_NO”
            ElementLongName “Section Number”
            ElementDataType “INTEGER”
            ElementKeyPosition “0”
            ElementLastRefreshDate “1996-02-01-12.00.00.000000”
            ElementLength “4”
            ElementNulls “F”
        END ELEMENT
        BEGIN ELEMENT
            Identifier
            "015"
            DateUpdated “1996-03-10”
            TimeUpdated “08.00.00”
            BriefDescription “Time and date section meets”
            ElementName “Time”
            ElementLongName “Semester OFFERINGS offered”
            ElementDataType “VARCHAR”
            ElementKeyPosition “0”
            ElementLastRefreshDate “1996-02-01-12.00.00.000000”
            ElementLength “9”
            ElementNulls “F”
        END ELEMENT
        BEGIN ELEMENT
            Identifier
            "016"
            DateUpdated “1996-03-10”
            TimeUpdated “08.00.00”
            BriefDescription “Building and room where section meets”
            ElementName “LOCATION”
            ElementLongName “Location”
```

```
        ElementDataType “VARCHAR”
        ElementKeyPosition “0”
        ElementLength “7”
        ElementNulls “F”
    END ELEMENT
    BEGIN ELEMENT
        Identifier  "017"
        DateUpdated “1996-03-10”
        TimeUpdated “08.00.00”
        BriefDescription “Faculty member assigned to teach the section in question”
        ElementName “FACULTY_SSS”
        ElementLongName “Professor”
        ElementDataType “VARCHAR”
        ElementKeyPosition “0”
        ElementLength “9”
        ElementNulls “F”
    END ELEMENT
END  RECORD
BEGIN RECORD
    Identifier  "018"
    DateUpdated “1996-03-10”
    TimeUpdated “08.00.00”
    BriefDescription  “Record describing faculty member”
    RecordDept_Name “FACULTY”
    RecordLongName “Faculty info”
    RecordLastRefreshDate “1996-02-01-12.00.00.000000”
    RecordType “TABLE”
    BEGIN ELEMENT
        Identifier
        "019"
        DateUpdated “1996-03-10”
        TimeUpdated “08.00.00”
        BriefDescription
         “Unique 4-char key identifying faculty member used in IMS database”
        ElementName “FACULTY_OLD_ID”
        ElementLongName “Faculty ID”
        ElementDataType “VARCHAR”
        ElementKeyPosition “1”
        ElementLastRefreshDate “1996-02-01-12.00.00.000000”
        ElementLength “4”
        ElementNulls “F”
    END ELEMENT
    BEGIN ELEMENT
        Identifier "020"
        DateUpdated “1996-03-10”
        TimeUpdated “08.00.00”
        BriefDescription
         “Unique 4-char key identifying faculty member used in IMS database”
        ElementName “FACULTY_SSN”
        ElementLongName “Social Security Number”
        ElementDataType “VARCHAR”
        ElementKeyPosition “1”
        ElementLastRefreshDate “1996-02-01-12.00.00.000000”
        ElementLength “9”
```

```
                ElementNulls “F”
            END ELEMENT
            BEGIN ELEMENT
                Identifier  "021"
                DateUpdated “1996-03-10”
                TimeUpdated “08.00.00”
                BriefDescription “Name of faculty member”
                ElementName “NAME”
                ElementLongName “Name”
                ElementDataType “VARCHAR”
                ElementKeyPosition “0”
                ElementLastRefreshDate “1996-02-01-12.00.00.000000”
                ElementLength “20”
                ElementNulls “F”
            END ELEMENT
            BEGIN ELEMENT
                Identifier  "022"
                DateUpdated “1996-03-10”
                TimeUpdated “08.00.00”
                BriefDescription “Address of faculty member”
                ElementName “ADDRESS”
                ElementLongName “Address”
                ElementDataType “VARCHAR”
                ElementKeyPosition “0”
                ElementLastRefreshDate “1996-02-01-12.00.00.000000”
                ElementLength “30”
                ElementNulls “F”
            END ELEMENT
            BEGIN ELEMENT
                Identifier
                 "023"
                DateUpdated “1996-03-10”
                TimeUpdated “08.00.00”
                BriefDescription “Rank of faculty member in question”
                ElementName “RANK”
                ElementLongName “Rank”
                ElementDataType “VARCHAR”
                ElementKeyPosition “0”
                ElementLastRefreshDate “1996-02-01-12.00.00.000000”
                ElementLength “7”
                ElementNulls “F”
            END ELEMENT
    END  RECORD
END DATABASE


COMMENT  Relationships defining join relationship between tables
BEGIN RELATIONSHIP
        Identifier
        "024"
        RelationshipName “Dept-Course“
        SourceObjectIdentifier "003"
        TargetObjectIdentifier "007"
        RelationshipType “EQUIVALENT”
        RelationshipOrdinality “1:N” [4]
        RelationshipBidirectional “T”
END RELATIONSHIP
BEGIN RELATIONSHIP
        Identifier
        "025"
        RelationshipName “Dept-Section “
        SourceObjectIdentifier "007"
        TargetObjectIdentifier "011"
        RelationshipType “EQUIVALENT”
        RelationshipOrdinality “1:N”
        RelationshipBidirectional “T”
END RELATIONSHIP
BEGIN RELATIONSHIP
        Identifier
        "026"
        RelationshipName “Course-Section“
        SourceObjectIdentifier "008"
        TargetObjectIdentifier "013"
        RelationshipType “EQUIVALENT”
        RelationshipOrdinality “1:N”
        RelationshipBidirectional “T”
END RELATIONSHIP
BEGIN RELATIONSHIP
        Identifier
        "027"
        DateCreated 1995-04-12
        TimeCreated 02.00.00
        RelationshipName "Faculty-Section"
        SourceObjectIdentifier "020"
        TargetObjectIdentifier "017"
        RelationshipType “EQUIVALENT”
        RelationshipOrdinality “1:1”
        RelationshipBidirectional “F”
END RELATIONSHIP
BEGIN RELATIONSHIP
        Identifier
        "028"
        DateCreated 1995-04-12
        TimeCreated 02.00.00
        RelationshipName "Faculty-Section"
        SourceObjectIdentifier "020"
        TargetObjectIdentifier "005"
        RelationshipType “EQUIVALENT”
        RelationshipOrdinality “1:1”
        RelationshipBidirectional “F”
END RELATIONSHIP
```

B.2 Representing hierarchical databases

[4] Note that the ordinality of this relationship is declared “1:N” since in the OFFERINGS table which lists the sections of each course taught there will be multiple instances of the COURSE_ID for every unique instance of the COURSE_ID in the COURSES table.

By definition, in a hierarchical data model, a record instance can have at most one physical owner. Each owning record, however, can have multiple instances of multiple types of children. The “root” of a hierarchical schema has no owner. The following IMS example is used to illustrate how to represent the parent-child relationship in a hierarchical schema. It is also used later in conjunction with the relational schema outlined below to illustrate the types of source-to-target relationships used to describe data interface programs (e.g., the program used to move data from operational systems to a data warehouse or to interface applications).

Instances of the Relationship object of RelationshipType “CONTAINS” are used to represent the parent-child relationships found in hierarchical databases. Instances of the Relationship object of RelationshipType “EQUIVALENT” are used to represent the equivalence in data values at different levels in a hierarchy.

The schema represented below is one thatmight be used to support a university scheduling system, where department is the root record type and there are two main branches of the hierarchy: course (with the children of type section and faculty). Assuming that this database resides on a host called “NEWTON” and has the owner “UADMIN,” the following is the MDIS specification:

```
COMMENT 1 DBD           NAME=USCHEDULE
COMMENT 2 SEGM          NAME=DEPT,BYTES=28
COMMENT 3 FIELD         NAME=(DNO,SEQ,BYTES=4,START=1
COMMENT 4 FIELD         NAME=DNAME,BYTES=20,START=5
COMMENT 5 FIELD         NAME=CHAIR,BYTE-4,START=26
COMMENT 6 SEGM          NAME=COURSE,PARENT=DEPT,BYTES=24
COMMENT 7 FIELD         NAME=CNO,BYTES=4,START=31
COMMENT 8 FIELD         NAME=CNAME,BYTES=20,START=36
COMMENT 9 SEGM          NAME=SECTION,PARENT=COURSE,BYTES=24
COMMENT 10       FIELD          NAME=SNO,BYTES=4,START=57
COMMENT 11       FIELD          NAME=SEMESTER,BYTE=1,START=63
COMMENT 12       FIELD          NAME=SDAY,BYTE=3,START=65
COMMENT 13       FIELD          NAME=STIME,BYTE=5,START=69
COMMENT 14       FIELD          NAME=SLOC,BYTE=7,START=75
COMMENT 15       FIELD          NAME=SFNO,BYTE=4,START=83
COMMENT 16       SEGM           NAME=FACULTY,PARENT=DEPT,BYTES=31
COMMENT 17       FIELD          NAME=FNO,BYTES=4,START=87
COMMENT 18       FIELD          NAME=FNAME, BYTES=20,START=92
COMMENT 19       FIELD          NAME=FTITLE,BYTES=7,START=112


BEGIN DEFINITION

COMMENT    Definition of MDIS model goes here....

END DEFINITION

BEGIN DATABASE
    Identifier "029"
    DateCreated “1992-12-02”
    TimeCreated “23.12.15”
    DateUpdated “1996-03-10”
    TimeUpdated “08.00.00”
    BriefDescription “IMS database containing department scheduling & faculty info”

        BEGIN ApplicationData
        Tool "TOOL XYZ
        BEGIN ToolAppData
     ACCESS-TYPE HDAM
        END ToolAppData
        END ApplicationData
    ServerName “NEWTON”
    DatabaseExtendedType "AIX1.0"
    OwnerName “UADMIN”
    DatabaseName “USCHEDULE”
    DatabaseStatus “PRODUCTION”
    DatabaseType “HIERARCHY”
COMMENT MDIS description of segments
    BEGIN RECORD
            Identifier "030"
            DateUpdated “1996-03-10”
            TimeUpdated “08.00.00”
            BriefDescription  “Record describing University department”
            RecordName “DEPT”
            RecordLongName “General department data”
            RecordLastRefreshDate “1996-02-01”
            RecordType “SEGMENT”
            BEGIN ELEMENT
                Identifier  "031"
                DateUpdated “1996-03-10”
                TimeUpdated “08.00.00”
                BriefDescription “Unique 4-char key identifying department”
                ElementName “DNO”
                ElementLongName “Department ID”
                ElementDataType “VARCHAR”
                ElementKeyPosition “1”
                ElementLastRefreshDate “1996-02-01”
                ElementLength “4”
                ElementNulls “F”
            END ELEMENT
            BEGIN ELEMENT
                Identifier "032"
                DateUpdated “1996-03-10”
                TimeUpdated “08.00.00”
                BriefDescription “Name of department”
                ElementName “DNAME”
                ElementLongName “Department name”
                ElementDataType “VARCHAR”
                ElementKeyPosition “0”
                ElementLastRefreshDate “1996-02-01”
                ElementLength “20”
                ElementNulls “F”
            END ELEMENT
            BEGIN ELEMENT
                Identifier  "033"
                DateUpdated “1996-03-10”
                TimeUpdated “08.00.00”
                BriefDescription “Faculty ID of individual currently serving as department
                    chair”
                ElementName “CHAIR”
                ElementLongName “Chairman ID”
                ElementDataType “VARCHAR”
                ElementKeyPosition “0”
                ElementLastRefreshDate “1996-02-01”
                ElementLength “4”
                ElementNulls “F”
            END ELEMENT
    END  RECORD
    BEGIN RECORD
            Identifier  "034"
            DateUpdated “1996-03-10”
            TimeUpdated “08.00.00”
            BriefDescription  “Record describing university course”
            RecordName “COURSE”
            RecordLongName “General course  data”
            RecordLastRefreshDate “1996-02-01”
            RecordType “SEGMENT”
            BEGIN ELEMENT
                Identifier "035"
                DateUpdated “1996-03-10”
                TimeUpdated “08.00.00”
                BriefDescription “Unique 4-char key identifying course”
                ElementName “COURSE_NO”
                ElementLongName “Course NO”
                ElementDataType “VARCHAR”
                ElementKeyPosition “1”
                ElementLastRefreshDate “1996-02-01”
                ElementLength “4”
                ElementNulls “F”
            END ELEMENT
            BEGIN ELEMENT
                Identifier  "036"
                DateUpdated “1996-03-10”
                TimeUpdated “08.00.00”
                BriefDescription “Name of course”
                ElementName “COURSE_NAME”
                ElementLongName “Course name”
                ElementDataType “VARCHAR”
                ElementKeyPosition “0”
                ElementLastRefreshDate “1996-02-01”
                ElementLength “20”
                ElementNulls “F”
            END ELEMENT
    END  RECORD
    BEGIN RECORD
            Identifier "037"
            DateUpdated “1996-03-10”
            TimeUpdated “08.00.00”
            BriefDescription  “Record describing section information”
```

```
RecordName “SECTION”
RecordLongName “General section data”
RecordLastRefreshDate “1996-02-01”
RecordType “SEGMENT”
BEGIN ELEMENT
    Identifier "038"
    DateUpdated “1996-03-10”
    TimeUpdated “08.00.00”
    BriefDescription “4-char id identifying section”
    ElementName “SNO”
    ElementLongName “Section ID”
    ElementDataType “VARCHAR”
    ElementKeyPosition “0”
    ElementLastRefreshDate “1996-02-01”
    ElementLength “4”
    ElementNulls “F”
END ELEMENT
BEGIN ELEMENT
    Identifier  "039"
    DateUpdated “1996-03-10”
    TimeUpdated “08.00.00”
    BriefDescription “Semester section is offered”
    ElementName “SEMESTER”
    ElementLongName “Semester section offered”
    ElementDataType “VARCHAR”
    ElementKeyPosition “0”
    ElementLastRefreshDate “1996-02-01”
    ElementLength “1”
    ElementNulls “F”
END ELEMENT
BEGIN ELEMENT
    Identifier  "040"
    DateUpdated “1996-03-10”
    TimeUpdated “08.00.00”
    BriefDescription “Days section is offered; values of TTH or MWF”
    ElementName “SDAY”
    ElementLongName “Days”
    ElementDataType “VARCHAR”
    ElementKeyPosition “0”
    ElementLastRefreshDate “1996-02-01”
    ElementLength “3”
    ElementNulls “F”
END ELEMENT
BEGIN ELEMENT
    Identifier "041"
    DateUpdated “1996-03-10”
    TimeUpdated “08.00.00”
    BriefDescription “Time section meets; military using colon”
    ElementName “STIME”
    ElementLongName “Time”
    ElementDataType “VARCHAR”
    ElementKeyPosition “0”
    ElementLastRefreshDate “1996-02-01”
    ElementLength “5”
```

```
            ElementNulls “F”
        END ELEMENT
        BEGIN ELEMENT
            Identifier  "042"
            DateUpdated “1996-03-10”
            TimeUpdated “08.00.00”
            BriefDescription “Building and room no where section meets”
            ElementName “SLOC”
            ElementLongName “Location”
            ElementDataType “VARCHAR”
            ElementKeyPosition “0”
            ElementLength “7”
            ElementNulls “F”
        END ELEMENT
        BEGIN ELEMENT
            Identifier  "043"
            DateUpdated “1996-03-10”
            TimeUpdated “08.00.00”
            BriefDescription “Faculty member assigned to teach section”
            ElementName “SFNO”
            ElementLongName “Professor”
            ElementDataType “VARCHAR”
            ElementKeyPosition “0”
            ElementLength “4”
            ElementNulls “F”
        END ELEMENT
END  RECORD
BEGIN RECORD
        Identifier "044"
        DateUpdated “1996-03-10”
        TimeUpdated “08.00.00”
        BriefDescription  “Record describing faculty member assigned to department”
        RecordName “FACULTY”
        RecordLongName “Faculty info”
        RecordLastRefreshDate “1996-02-01”
        RecordType “SEGMENT”
        BEGIN ELEMENT
            Identifier "045"
            DateUpdated “1996-03-10”
            TimeUpdated “08.00.00”
            BriefDescription “Unique 4-char key identifying faculty member”
            ElementName “FNO”
            ElementLongName “Faculty ID”
            ElementDataType “VARCHAR”
            ElementKeyPosition “1”
            ElementLastRefreshDate “1996-02-01”
            ElementLength “4”
            ElementNulls “F”
        END ELEMENT
        BEGIN ELEMENT
            Identifier  "046"
            DateUpdated “1996-03-10”
            TimeUpdated “08.00.00”
            BriefDescription “Name of faculty member”
            ElementName “FNAME”
            ElementLongName “Name”
            ElementDataType “VARCHAR”
            ElementKeyPosition “0”
            ElementLastRefreshDate “1996-02-01”
            ElementLength “20”
            ElementNulls “F”
        END ELEMENT
        BEGIN ELEMENT
            Identifier  "047"
            DateUpdated “1996-03-10”
            TimeUpdated “08.00.00”
            BriefDescription “Rank of faculty member in question”
            ElementName “FTITLE”
            ElementLongName “Rank”
            ElementDataType “VARCHAR”
            ElementKeyPosition “0”
            ElementLastRefreshDate “1996-02-01”
            ElementLength “7”
            ElementNulls “F”
        END ELEMENT
    END  RECORD
END DATABASE


COMMENT  Relationships defining parent-child relationships
BEGIN RELATIONSHIP
        Identifier
        "048"
        DateCreated 1992-12-02
        TimeCreated 23.12.15
        BriefDescription “Defines relationship between department and course records”
        RelationshipName “Dept-Course“
        SourceObjectIdentifier "034"
        TargetObjectIdentifier "037"
        RelationshipType “CONTAINS”
        RelationshipOrdinality “1:N”
        RelationshipBidirectional “F”
END RELATIONSHIP

BEGIN RELATIONSHIP
        Identifier
        "049"
        DateCreated 1992-12-02
        TimeCreated 23.12.15
        BriefDescription “Defines relationship between department course and section
                records”
        RelationshipName “Course-Section“
        SourceObjectIdentifier "030"
        TargetObjectIdentifier "040"
        RelationshipType “CONTAINS”
        RelationshipOrdinality “1:N”
        RelationshipBidirectional “F”
END RELATIONSHIP
```

```
BEGIN RELATIONSHIP
        Identifier
        "050"
        DateCreated 1992-12-02
        TimeCreated 23.12.15
        BriefDescription “Defines relationship between dept & faculty records”
        RelationshipName “Dept-faculty“
        SourceObjectIdentifier "030"
        TargetObjectIdentifier "040"
        RelationshipType “CONTAINS”
        RelationshipOrdinality “1:N”
        RelationshipBidirectional “F”
END RELATIONSHIP

COMMENT Relationships defining equivalent data values
BEGIN RELATIONSHIP
        Identifier
        "051"
        DateCreated 1992-12-02
        TimeCreated 23.12.15
        BriefDescription “Defines relationship between data elements dept.chair  &
                faculty.fno”
        RelationshipName “Chair=Fno “
        SourceObjectIdentifier “NEWTON.UADMIN.USCHEDULE.DEPT.CHAIR”
        TargetObjectIdentifier “NEWTON.UADMIN.USCHEDULE.FACULTY.FNO”
        RelationshipType “EQUIVALENT”
        RelationshipOrdinality “1:1”
        RelationshipBidirectional “T”
END RELATIONSHIP

BEGIN RELATIONSHIP
        Identifier
        "052"
        DateCreated 1992-12-02
        TimeCreated 23.12.15
        BriefDescription “Defines relationship between data elements faculty.fno &
                section.sno”
        RelationshipName “Chair=Fno “
        SourceObjectIdentifier "033"
        TargetObjectIdentifier "043"
        RelationshipType “EQUIVALENT”
        RelationshipOrdinality “1:1”
        RelationshipBidirectional “T”
END RELATIONSHIP
```

B.3 Representing files

Th MDIS representation of the following COBOL copybook illustrates three features commony found in the descriptions of files:

- The repetition of multiple instances of a data element within a single instance of a record (e.g., COBOL OCCURS clauses). The property ElementOrdinality on the Element object is used to represent the number of legal instances that can occur.

- A group of contiguous data elements within a record which can be referred to by a single logical name (e.g., GROUPs in COBOL). This type of structue is handed by specifying an element within the 01 record definition with an ElementName of the <GROUP name> and an ElementDataType of type “RECORD” (to indicate that it refers to a set of two or more continguous data elements). A Record definition is then specified in the MDIS file with a RecordName of < GROUP name>.
- The ability to repartition a previously specified set of contiguous data elements (e.g., REDEFINES in COBOL). In this case, a data element for the original data element appears in the record definition in the appropriate place, a second record definition is specified using the REDEFINE name as RecordName and defining the subcomponents of the RDEFINE as Elements within that record definition. An instance of the MDIS Relationship object is created with RelationshipType “REDEFINES” to specify the relationship between the two.
- The way Cobol Groups are modelled in the MDIS specification is through Elements with the dataype of "RECORD" and a corresponding Record of type "GROUP"; and a Relationship of type "CONTAINS" between the parent record and its group. For version 1.0, this relationship was specified between the two records and the type of relationship was of type "CONTAINS". In order to overcome the problem if nested groups have the samm e name, for Version 1.1 this relationship iis modelled betgween the Element of type "RECORD" and the Record of type "GROUP". The type of relationship is "GROUP-EQUIVALENT".

```
COMMENT      FD  CUSTOMER-ORDER-RECORD
COMMENT      LABEL RECORDS ARE OMITTED.
COMMENT      01  CUSTOMER-RECORD.
COMMENT      03  SOCIAL-SECURITY-NUMB      PIC X(11).
COMMENT      03  CUSTOMER-NAME             PIC X(40).
COMMENT      03  CUSTOMER-ADDRESS.
COMMENT          05  STREET-ADDRESS-1      PIC X(30).
COMMENT          05  STREET-ADDRESS-2      PIC X(30).
COMMENT          05  CITY                  PIC X(28).
COMMENT          05  STATE                 PIC XX.
COMMENT          05  ZIP-CODE              PIC X(10).
COMMENT      03  CUSTOMER-PHONE            PIC X(12) OCCURS 2 TIMES.
COMMENT
COMMENT      01  ORDERS-RECORD.
COMMENT      03  ORDER-NUMBER              PIC 9(8)     COMP.
COMMENT      03  STOCK-NUMBER              PIC 9(8)     COMP.
COMMENT      03  FABRIC-CHARGE             PIC S9(13)V99 COMP-3.
COMMENT      03  FABRIC-CHARGE-2      REDEFINES FABRIC-CHARGE.
COMMENT          05  FABRIC-DOLLARS        PIC S9(13).
COMMENT          05  FABRIC-CENTS          PIC 99.
COMMENT
```

```
BEGIN HEADER
   CharacterSet "ENGLISH"
```

```
    ExportingTool "XYZ"
    ToolVersion  "V3.1"
    MDISVerson "1.0"
    Date "1996-05-08"
    Time "22.46.14"
END HEADER

BEGIN DEFINITION

COMMENT    Definition of MDIS model goes here....

END DEFINITION

BEGIN DATABASE
   Identifer      "053"
   ServerName      "SERVER1"
   OwnerName       "MDC"
   DatabaseName    "CUSTOMER-ORDER-RECORD"
   DatabaseType    "FILE"
   DateCreated     "1996-05-08"
   BriefDecsription "COPYBOOK defining format of file CSP101"

   BEGIN RECORD
      Identifier "054"
      RecordName      "CUSTOMER-RECORD"
      RecordLongName  "Customer"
      RecordType      "RECORD"

      BEGIN ELEMENT
         Identifier "055"
         ElementName        "SOCIAL-SECURITY-NUMB"
         ElementLongName     "Customer Social Security Number"
         ElementDataType     "CHAR"
         ElementKeyPosition  "0"
         ElementLength       "11"
         ElementNulls        "T"
      END ELEMENT

      BEGIN ELEMENT
         Identifier "056"
         ElementName        "CUSTOMER-NAME"
         ElementLongName     "Customer Name"
         ElementDataType     "CHAR"
         ElementKeyPosition "0"
         ElementLength       "40"
         ElementNulls        "T"
      END ELEMENT

BEGIN ELEMENT
         Identifier "057"
         ElementName        "CUSTOMER-ADDRESS"
         ElementLongName     "Customer Address"
         ElementDataType     "RECORD"
         ElementKeyPosition  "0"
         ElementLength      “100”
         ElementNulls      "T"
      END ELEMENT


      BEGIN ELEMENT
         Identifier "058"
         ElementName     "CUSTOMER-PHONE"
         ElementLongName "Customer Phone"
         ElementDataType "CHAR"
         ElementLength  “12”
         ElementKeyPosition  “0”
         ElementNulls    "T"
         ElementOrdinality  “2”
      END ELEMENT
   END RECORD

BEGIN RECORD
      Identifier "059"
      RecordName      "CUSTOMER-ADDRESS"
      RecordLongName  "Customer Address"
      RecordType     "GROUP"

      BEGIN ELEMENT
         Identifier "060"
         ElementName     "STREET-ADDRESS-1"
         ElementLongName "Street Address 1"
         ElementDataType "CHAR"
         ElementLength   “30”
         ElementKeyPosition “0”
         ElementNulls    "T"
      END ELEMENT

      BEGIN ELEMENT
         Identifier "061"
         ElementName     "STREET-ADDRESS-2"
         ElementLongName "Street Address 2"
         ElementDataType "CHAR"
         ElementLength   “30”
         ElementKeyPosition  “0”
         ElementNulls    "T"
      END ELEMENT

      BEGIN ELEMENT
         Identifier "062"
         ElementName     "CITY"
         ElementLongName "City"
         ElementDataType "CHAR"
         ElementLength   “28”
         ElementKeyPosition  “0”
         ElementNulls    "T"
      END ELEMENT

      BEGIN ELEMENT
```

```
      Identifier "063"
      ElementName     "STATE"
      ElementLongName "State"
      ElementDataType "CHAR"
      ElementLength   “2”
      ElementKeyPosition  “0”
      ElementNulls    "T"
   END ELEMENT

   BEGIN ELEMENT
      Identifier "064"
      ElementName     "ZIP-CODE"
      ElementLongName "Zip-Code"
      ElementDataType "CHAR"
      ElementLength   “10”
      ElementKeyPosition “0”
      ElementNulls    "T"
   END ELEMENT

END RECORD

BEGIN RECORD
   Identifier  "065"
   RecordName      "ORDER-RECORD"
   RecordLongName  "Order"
   RecordType      “FILE”

   BEGIN ELEMENT
      Identifier  "066"
      ElementName     "ORDER-NUMBER"
      ElementLongName "Order Number"
      ElementDataType "INTEGER"
      ElementLength  “8”
      ElementKeyPosition  “1”
      ElementNulls    "F"
   END ELEMENT

   BEGIN ELEMENT
      Identifier "067"
      ElementName     "STOCK-NUMBER"
      ElementLongName "Stock Number"
      ElementDataType "INTEGER"
      ElementLength  “8”
      ElementKeyPosition “0”
      ElementNulls    "T"
   END ELEMENT

   BEGIN ELEMENT
      Identifier "068"
      ElementName     "FABRIC-CHARGE"
      ElementLongName "Fabric Charge"
      ElementDataType "DECIMAL"
      ElementLength  “13”
      ElementPrecision    "2"
```

```
         ElementKeyPosition  “0”
         ElementNulls    "T"
      END ELEMENT

   END RECORD

COMMENT
COMMENT     REDEFINES statement.
COMMENT
   BEGIN RECORD
      Identifier "069"
      RecordName      "FABRIC-CHARGE-2"
      RecordLongName  "Fabric Charge Redefined"
      RecordType      "GROUP"

      BEGIN ELEMENT
         Identifier "070"
         ElementName     "FABRIC-DOLLARS"
         ElementLongName "Fabric Charge Dollars"
         ElementDataType "INTEGER"
         ElementLength   “13”
      END ELEMENT

      BEGIN ELEMENT
         Identifier "071"
         ElementName     "FABRIC-CENTS"
         ElementLongName "Fabric Chg Cents"
         ElementDataType "INTEGER"
         ElementLength   “2”
      END ELEMENT

   END RECORD



BEGIN RELATIONSHIP
      Identifier "072"
         SourceObjectIdentifier "069"
      TargetObjectIdentifier "068"
      RelationshipType    "REDEFINES"
      RelationshipOrdinality  "1:1"
      RelationshipBiDirectional       "T"
   END RELATIONSHIP

   BEGIN RELATIONSHIP
      Identifier "073"
      TargetObjectIdentifier "057"
      SourceObjectIdentifier "059"
      RelationshipType    "GROUP-EQUIVALENT"
      RelationshipOrdinality  "1:1"
      RelationshipBiDirectional       "F"
   END RELATIONSHIP

END DATABASE
```

B.4 Representing network databases

The network data model allows multiple paths to the same record type. Instances of the Relationship object of RelationshipType “LINK-TO” are used to define the set relationships supported by network databases.

```
COMMENT
COMMENT    Here is a network schema example...
COMMENT
COMMENT
COMMENT SCHEMA NAME IS EMPLOYEES-AND-DEPTS
COMMENT
COMMENT RECORD NAME IS EMPLOYEE;
COMMENT    DUPLICATES ARE NOT ALLOWED
COMMENT            FOR EMPID IN EMPLOYEE.
COMMENT       EMPID   ; TYPE IS CHARACTER.
COMMENT       ENAME   ; TYPE IS CHARACTER.
COMMENT       STATUS  ; TYPE IS FIXED DECIMAL.
COMMENT
COMMENT RECORD NAME IS DEPT;
COMMENT    DUPLICATES ARE NOT ALLOWED
COMMENT            FOR DEPTNO IN DEPT.
COMMENT       DEPTNO  ; TYPE IS CHARACTER.
COMMENT       DNAME   ; TYPE IS CHARACTER.
COMMENT
COMMENT SET NAME IS DEPT-EMP;
COMMENT    OWNER IS DEPT;
COMMENT    ORDER IS SORTED BY DEFINED KEYS
COMMENT       DUPLICATES ARE NOT ALLOWED.
COMMENT    MEMBER IS EMPLOYEE;
COMMENT       INSERTION IS AUTOMATIC
COMMENT       RETENTION IS MANDATORY;
COMMENT       KEY IS ASCENDING EMPID IN EMPLOYEE;
COMMENT       SET SELECTION IS BY VALUE OF DEPTNO IN DEPT.
COMMENT


BEGIN HEADER
   CharacterSet "ENGLISH"
   ExportingTool "XYZ"
   ToolVersion  "V3.1"
   MDISVerson "1.0"
   Date "1996-05-08"
   Time "22.46.14"
END HEADER

BEGIN DEFINITION

COMMENT    Definition of MDIS model goes here....

END DEFINITION
```

```
BEGIN DATABASE
   Identifier    "074"
   ServerName     "SERVER1"
   OwnerName      "MDC"
   DatabaseName “EMPLOYEES-AND-DEPTS”
   DatabaseType   "NETWORK"
   DateCreated    "1996-05-08"
   BriefDecsription “IDMS database defining organizational structure”


   BEGIN RECORD
      Identifier    "075"
      RecordLongName  "Employees"
      Recordname "EMPLOYEE"
      RecordType     “RECORD”

      BEGIN ELEMENT
         Identifier "076"
         ElementName "EMPID"
         ElementLongName    "Employee ID"
         ElementDataType    "CHAR"
         ElementKeyPosition  “1”
         ElementLength      “5”
         ElementNulls       "F"
      END ELEMENT

      BEGIN ELEMENT
         Identifier "077"
         ElementName "ENAME"
         ElementLongName    "Employee Name"
         ElementDataType    "CHAR"
         ElementKeyPosition  “0”
         ElementLength      “20”
         ElementNulls       "F"
      END ELEMENT

      BEGIN ELEMENT
         Identifier "078"
         ElementLongName    "Employee Status"
         ElementName "Status"
         ElementDataType    "INTEGER"
         ElementKeyPosition  “0”
         ElementLength      “3”
         ElementNulls       "F"
      END ELEMENT

   END RECORD


   BEGIN RECORD
      Identifier  "079"
      RecordLongName  "Department"
      RecordType     RECORD
      RecordName "DEPT"
```

```
        BEGIN ELEMENT
          Identifier  "080"
          ElementName "DEPTNO"
          ElementLongName    "Department Number"
          ElementDataType    "CHAR"
          ElementKeyPosition  “1”
          ElementLength      “6”
          ElementNulls       "F"
        END ELEMENT

        BEGIN ELEMENT
          Identifier  "081"
          ElementName "DNAME"
          ElementLongName    "Department Name"
          ElementDataType    "CHAR"
          ElementKeyPosition  “0”
          ElementLength      “20”
          ElementNulls       "F"
        END ELEMENT

    END RECORD

    BEGIN RELATIONSHIP
        Identifier  "082"
        SourceObjectIdentifier    "079"
        TargetObjectIdentifier    "075"
        RelationshipBidirectional      "T"
        RelationshipOrdinality         "1:N"
        RelationshipType    "LINKTO"
    END RELATIONSHIP

END DATABASE
```

B.5 Representing object-oriented databases

Object-oriented databases require that one represent the following types of concepts:

- The fact that the properties of a subclass are inherited by any superclass identified in its definition. An instance of the Relationship object of RelationshipType “INHERITS-FROM” is used to define this type of relationship. In other words, the class hierarchy is not represented by nesting the definitions of the Record objects representing subclasses within instances of instances of Record objects representing superclasses since this would violate the constraint that objects cannot be nested within objects of the same type. (Otherwise in the case that an object had more than one ancestor, multiple copies of that object definition would appear within the MDIS file.) Instead, the Relationship object is used to represent class hierarchies.
- Properties of a Record of RecordType “CLASS” can refer to the value of one or more object identifiers. These are represented in the Element object by declaring an ElementDataType of “POINTER.”

```
COMMENT
COMMENT    THE    FOLLOWING    MDIS    ILLUSTRATION    REPRESENTS    A
DATABASE
COMMENT  WHICH USES THE OBJECT-ORIENTED MODEL
COMMENT class SOFTWARE
COMMENT    {
COMMENT    public:
COMMENT              virtual void DELETE ()
COMMENT              {
COMMENT               // function code for DELETE
COMMENT               }
COMMENT    char* GET_NAME();
COMMENT              {
COMMENT               // function code for GET_NAME
COMMENT              }
COMMENT    void  SET_NAME(char* name);
COMMENT               {
COMMENT               //  function code for SET_NAME
COMMENT              }
COMMENT    private:
COMMENT    char* NAME;
COMMENT
COMMENT }  // end SOFTWARE class
COMMENT
COMMENT class PACKAGE : public SOFTWARE
COMMENT {
COMMENT    public:
COMMENT              void DELETE ()
COMMENT              {
COMMENT               // overload function code for DELETE
COMMENT               }
COMMENT    private:
COMMENT              void ACCESS ()
COMMENT              {
COMMENT               // function code for ACCESS
COMMENT              }
COMMENT }  // end PACKAGE class
COMMENT
COMMENT class CUSTOM : public SOFTWARE
COMMENT {
COMMENT     public:
COMMENT
COMMENT } // end CUSTOM class

BEGIN HEADER
   CharacterSet "ENGLISH"
   ExportingTool "OBJECT"
   ToolVersion  "V3.1"
   MDISVerson "1.0"
   Date "1996-05-08"
   Time "22.46.14"
END HEADER
```

```
BEGIN DEFINITION
   Definition of MDIS model
END DEFINITION

BEGIN DATABASE
    Identifier "083.5"
    ServerName “SERVER”
     DatabaseExtendedType "DATABASE1.0"
    OwnerName  “MDC”
    DatabaseName “OBJDB”
    BriefDescription “C++ database defining software”
    DatabaseType "OBJECT"

BEGIN RECORD
   Identifier "083"
   BriefDescription  "Class of Software Objects"

         BEGIN ApplicationData
         Tool "ObjectStore"
         BEGIN ToolAppData
         OMG_CLASS LOAD
         END ToolAppData
         END ApplicationData
   RecordName "SOFTWARE"
   RecordLongName "Software Class"
   RecordType "CLASS"

BEGIN ELEMENT
      Identifier "084"
      ElementName "NAME"
      ElementLongName  "Software Package Name"
      ElementDataType "VARCHAR"
      ElementOrdinality "1"
      ElementLength "30"
      ElementNulls "T"
END ELEMENT

BEGIN ELEMENT
      Identifier "085"
      ElementName "DELETE METHOD"
      ElementLongName  "Software Delete Method"
 BEGIN ApplicationData
         Tool "Smalltalk"
         BEGIN ToolAppData
         Declare Delete Function
         END ToolAppData
         END  ApplicationData
      ElementDataType "PROGRAM"
      ElementOrdinality "1"
      ElementLength "25000"
      ElementNulls "T"
END ELEMENT

END RECORD
```

```
BEGIN RECORD
   Identifier "086"
   BriefDescription "Subclass containing custom software applications"
   BEGIN ApplicationData
         Tool "Versant"
         BEGIN ToolAppData
         Inherit Method Default
         END ToolAppData
    END ApplicationData
   RecordName "CUSTOM"
   RecordLongName "Custom Software"
   RecordType "CLASS"
END RECORD

BEGIN RECORD
   Identifier "087"
   BriefDescription "Subclass containing packaged software applications"
   BEGIN ApplicationData
         Tool "Versant"
         BEGIN ToolAppData
         Inherit Method Default
         END ToolAppData
   END ApplicationData
   RecordName "PACKAGE"
   RecordLongName "Package Software"
   RecordType "CLASS"

BEGIN ELEMENT
     Identifier "088"
     ElementName "DELETE METHOD"
     ElementLongName  "Software Delete Method"

     BEGIN ApplicationData
         Tool "Versant"
         BEGIN ToolAppData
         Inherit Method Default
         END ToolAppData
      END ApplicationData
     ElementDataType "PROGRAM"
     ElementOrdinality "1"
     ElementLength "40000"
     ElementNulls "T"
END ELEMENT

BEGIN ELEMENT
     Identifier "089"
     ElementName "ACCESS METHOD"
     ElementLongName  "Software Access Method"
     ElementDataType "PROGRAM"
     ElementOrdinality "1"
     ElementLength "40000"
     ElementNulls "T"
END ELEMENT
END RECORD
END DATABASE

COMMENT  ***  Software Class to Package Subclass  ***

BEGIN RELATIONSHIP
    Identifier "100"
    SourceObjectIdentifier "087"
    TargetObjectIdentifier "083"
    RelationshipType "INHERITS-FROM"
    RelationshipOrdinality “1:N”
    RelationshipBidirectional “F”
END RELATIONSHIP

BEGIN RELATIONSHIP
    Identifier "101"
     SourceObjectIdentifier "086"
    TargetObjectIdentifier "083"

    RelationshipType "INHERITS-FROM"
    RelationshipOrdinality “1:N”
    RelationshipBidirectional “F”
END RELATIONSHIP
```

B.6 Representing multi-dimensional databases

A muti-dimensional database has the following characteristics:

- It consists of one or more dimensions, each of which consists of a hierarchy of members. Each dimension is represented by an instance of the MDIS Dimension object (e.g., Product in the example below). Dimension members (e.g., Colas in the example below) are defined using the Element object.
- Members can belong to more than one hierarchy within a dimension. Instances of the Relationship object with a RelationshipType of “CONTAINS” are used to indicate which members are at a given level of detail or aggregation within the hierarchy.

```
COMMENT
COMMENT
COMMENT
COMMENT <Gen 1
COMMENT Product
COMMENT                      <SPARSE
COMMENT   <Gen 2
COMMENT   "100"
COMMENT <ALTNAME Colas
COMMENT <Gen 3
COMMENT "100-10"
COMMENT <ALTNAME Cola
COMMENT <Gen 3
COMMENT "100-20"
COMMENT <ALTNAME "Diet Cola"
COMMENT <Gen 3
COMMENT "100-30"
COMMENT <ALTNAME "Caffeine Free Cola"
COMMENT <Gen 2
COMMENT "200"
COMMENT <ALTNAME "Root Beer"
COMMENT <Gen 3
COMMENT "200-10"
COMMENT <ALTNAME "Old Fashioned"
COMMENT <Gen 3
COMMENT "200-20"
COMMENT <ALTNAME "Diet Root Beer"
COMMENT <Gen 3
COMMENT "200-30"
COMMENT <ALTNAME Sasparilla
COMMENT <Gen 3
COMMENT "200-40"
COMMENT <ALTNAME "Birch Beer"
COMMENT <Gen 2
COMMENT "300"
COMMENT <ALTNAME "Cream Soda"
COMMENT <Gen 3
COMMENT "300-10"
COMMENT <ALTNAME "Dark Cream"
COMMENT <Gen 3
COMMENT "300-20"
COMMENT <ALTNAME "Vanilla Cream"
COMMENT <Gen 3
COMMENT "300-30"
COMMENT <ALTNAME "Diet Cream"
COMMENT <Gen 2
COMMENT "400"
COMMENT <ALTNAME "Fruit Soda"
COMMENT <Gen 3
COMMENT "400-10"
COMMENT <ALTNAME Grape
COMMENT <Gen 3
COMMENT "400-20"
COMMENT <ALTNAME Orange
COMMENT <Gen 3
COMMENT "400-30"
COMMENT <ALTNAME Strawberry
COMMENT <Gen 2
COMMENT Diet
COMMENT <ALTNAME "Diet Drinks"
COMMENT <UNARY ~
COMMENT <Gen 3
COMMENT "100-20"
COMMENT <Gen 3
COMMENT "200-20"
COMMENT <Gen 3
COMMENT "300-30"
COMMENT
COMMENT

BEGIN HEADER
    CharacterSet "ENGLISH"
    ExportingTool "ARB"
    ToolVersion "V3.1"
    MDISVerson "1.0"
    Date "1996-05-07"
    Time "22.46.14"
 END HEADER

BEGIN DEFINITION
   Definition of MDIS model
END DEFINITION

BEGIN DATABASE
    Identifier "102"
    ServerName “SERVER”
    OwnerName  “OLAP”
    DatabaseName “ARBOR”
    BriefDescription “OLAP Example of multi-dimensional database”
    DatabaseType "MULTIDIMENSIONAL"

BEGIN DIMENSION
   Identifier "090"
   BriefDescription "Consumable items which company makes"
    BEGIN ApplicationData
          Tool "Essbase V3.1"
          BEGIN ToolAppData
                    MAX-NUM-DIM 5 GENERATION 1-Based
          END ToolAppData
    END ApplicationData
LEVEL 0-Based   UNARY {~, +, -, *, /, %} >"
   DimensionLongName "Products"
   DimensionName "PRODUCT"
   DimensionType "SPARSE
   DimensionCount “18”

BEGIN ELEMENT
   Identifier "091"
   ElementName "100"
   ElementLongName "Colas"
   ElementOrdinality "1"
   ElementDataType "VARCHAR"
END ELEMENT

BEGIN ELEMENT
   Identifier "092"
   ElementName "200"
   ElementLongName "Root Beer"
   ElementOrdinality "1"
   ElementDataType "VARCHAR"
END ELEMENT

BEGIN ELEMENT
   Identifier "093"
   ElementName "100-20"
   ElementLongName "DietCola"
   ElementOrdinality "1"
   ElementDataType "VARCHAR"
END ELEMENT

BEGIN ELEMENT
   Identifier  "094"
   ElementName  "200-10"
   ElementLongName "Old Fashioned"
   ElementOrdinality "1"
   ElementDataType "VARCHAR"
END ELEMENT

BEGIN LEVEL
   Identifier "095"
   LevelName “BASE”
   LevelNumber "0"
   LevelType "NAME-LEVEL"
END LEVEL

BEGIN LEVEL
   Identifier "096"
   LevelName  “FIRST”
   LevelNumber "1"
   LevelType "NAME-LEVEL"
END LEVEL

BEGIN LEVEL
 Identifier "097"
 LevelName "BEVERAGES"
 LevelType "NAME-LEVEL"
END LEVEL

END DIMENSION

END DATABASE

COMMENT  *** HIERARCHY RELATIONSHIPS ***

BEGIN RELATIONSHIP
    Identifier "151"
    SourceObjectIdentifier "090"
    TargetObjectIdentifier "091"
    RelationshipType "CONTAINS"
    RelationshipOrdinality “1:N”
    RelationshipBidirectional “F”
```

```
END RELATIONSHIP

BEGIN RELATIONSHIP
    Identifier"098"
    SourceObjectIdentifier "090"
    TargetObjectIdentifier "092"
    RelationshipType "CONTAINS"
    RelationshipOrdinality “1:N”
    RelationshipBidirectional “F”

END RELATIONSHIP

BEGIN RELATIONSHIP
    Identifier "099"
    SourceObjectIdentifier"091"
    TargetObjectIdentifier"093"
    RelationshipType "CONTAINS"
    RelationshipOrdinality “1:N”
    RelationshipBidirectional “F”
END RELATIONSHIP

BEGIN RELATIONSHIP
    Identifier "100"
    SourceObjectIdentifier "092"
    TargetObjectIdentifier "094"
    RelationshipType "CONTAINS"
    RelationshipOrdinality “1:N”
    RelationshipBidirectional “F”

END RELATIONSHIP

COMMENT  *** BASE LEVEL RELATIONSHIPS ***



BEGIN RELATIONSHIP
    Identifier "101"
     SourceObjectIdentifier "095"
    TargetObjectIdentifier"094"
     RelationshipType "INCLUDES"
    RelationshipOrdinality "1:N"
    RelationshipBidirectional "F"
END RELATIONSHIP

BEGIN RELATIONSHIP
    Identifier "102"
    SourceObjectIdentifier "095"
    TargetObjectIdentifier "093"
    RelationshipType "INCLUDES"
    RelationshipOrdinality "1:N"
    RelationshipBidirectional "F"

END RELATIONSHIP
```

```
COMMENT  *** FIRST LEVEL RELATIONSHIPS ***

BEGIN RELATIONSHIP
    Identifier "103"
    SourceObjectIdentifier "096"
    TargetObjectIdentifier "091"
    RelationshipType "INCLUDES"
    RelationshipOrdinality "1:N"
    RelationshipBidirectional "F"
END RELATIONSHIP

BEGIN RELATIONSHIP
    Identifier "104"
    SourceObjectIdentifier "096"
    TargetObjectIdentifier "092"
    RelationshipType "INCLUDES"
    RelationshipOrdinality "1:N"
    RelationshipBidirectional "F"
END RELATIONSHIP

COMMENT  *** NAMED LEVEL RELATIONSHIPS ***

BEGIN RELATIONSHIP
  Identifier
"105"
  SourceIdentifier  "097"
  TargetIdentifier   "092"
  RelationshipType "INCLUDES"
  RelationshipOrdinality "1:N"
END RELATIONSHIP

BEGIN RELATIONSHIP
  Identifier "106"
  SourceIdentifier  "097"
  TargetIdentifier   "091"
  RelationshipType "INCLUDES"
  RelationshipOrdinality "1:N"
END RELATIONSHIP
```

B.7 Representing inter-database relationships

If one assumed that the data values found in the IMS USCHEDULE database were used to populate the data values found in the relational COURSE_CATALOG (where possible), two types of Relationship objects would be used to define the inter-database relationships:

- Instances of RelationshipType “EQUIVALENT” to indicate the equivalence between such fields as DNO and DEPT_ID, CNO and COURSE_NO, and
- Instances of RelationshipType “DERIVED” to indicate that:

> The TIME field in the OFFERINGS table consists of STIME concatenated with SDAY with an intervening blank

Example of bidirectional inter-database equivalence relationship

```
BEGIN RELATIONSHIP
        Identifier
        "107"
        RelationshipName “IMS Dept-RDB Dept“
        SourceObjectIdentifier "031"
        TargetObjectIdentifier "003"
        RelationshipType “EQUIVALENT”
        RelationshipOrdinality “1:1”
        RelationshipBidirectional “T”
END RELATIONSHIP
```

Example of inter-database equivalence relationship which is not bidirectional

```
BEGIN RELATIONSHIP
        Identifier
        "108"
        RelationshipName “IMS Dept-RDB Dept“
        SourceObjectIdentifier ????
        TargetObjectIdentifier ????
        RelationshipType “EQUIVALENT”
        RelationshipOrdinality “1:1”
        RelationshipBidirectional “F”
END RELATIONSHIP
```

Example of derived relationship

```
BEGIN RELATIONSHIP
        Identifier
        "109"
        RelationshipName “IMS DAY & TIME -RDB TIME“
        SourceObjectIdentifier "041"
        TargetObjectIdentifier "015"
        SourceSequenceOrder  “ 1:2”
        RelationshipExpression “Concatenate IMS.SECTION.STIME with
        IMS.SECTION.SDAY”
        RelationshipType “DERIVED”
        RelationshipOrdinality “N:N”
        RelationshipBidirectional “F”
END RELATIONSHIP
BEGIN RELATIONSHIP
        Identifier
        "110"
        RelationshipName “IMS DAY & TIME -RDB TIME“
        SourceObjectIdentifier "040"
        TargetObjectIdentifier "015"
        SourceSequenceOrder  “ 2:2”
        RelationshipExpression “Concatenate IMS.SECTION.STIME
        IMS.SECTION.SDAY”
        RelationshipType “DERIVED”
        RelationshipOrdinality “N:N”
        RelationshipBidirectional “F”
```

```
END RELATIONSHIP
```

Example of Relationship of RelationshipType "DERIVED":

The following is pseudocode that represents a derived relationship computed from the source parts *cobolfs.s2k.adpinfo.XYZ-CONVERSION-INFO.CLIENT-NUMBER* and *cobolfs.s2k.adptape.RECORD-100.HOME-DEPT-NO*.

> Compute the target value A1ORGANIZATION-1-CODE from the source parts. If cobolfs.s2k.adpinfo.XYZ-CONVERSION-INFO.CLIENT-NUMBER is the same as the string value AAE Move a partial field cobolfs.s2k.adptape.RECORD-100.HOME-DEPT-NO starting in position 1 for a length of 2 Otherwise If cobolfs.s2k.adpinfo.XYZ-CONVERSION-INFO.CLIENT-NUMBER is the same as the string value AAG the target string is the string value 00 . Otherwise If cobolfs.s2k.adpinfo.XYZ-CONVERSION-INFO.CLIENT-NUMBER is the same as the string value AAC the target string is the string value 01 . Otherwise the target string is SPACES .

This example would require something like the following two relationship instances:

```
BEGIN RELATIONSHIP
        Identifier "010"
        DateCreated “1992-12-02”
        TimeCreated “23.12.15”
        BriefDescription “Defines means of computing the Organization Code ”
        BEGIN ApplicationData
                Tool "XYZ"
                BEGIN ToolAppData
                        ORDINALITY 2:1
                END ToolAppData
        END ApplicationData
        RelationshipName "OrgCode1"
        SourceObjectIdentifier "3000"
        TargetObjectIdentifier "4000"
        SourceSequenceOrder “1:2”
        RelationshipExpression “if (stringequal %1 “AAE”) return(substring
        (%2,1,3) else if  (stringequal %1 “AAG”) return(“00”) else if (stringequal
        %1 “AAC”) return (“01”) else return (“  “);”
        RelationshipType “DERIVED”
        RelationshipOrdinality "2:1"
        RelationshipBidirectional "T"
END RELATIONSHIP

BEGIN RELATIONSHIP
        Identifier "011"
        DateCreated "1992-12-02"
        TimeCreated "23.12.15"
        BriefDescription “Defines means of computing the Organization Code ”
        BEGIN ApplicationData
                Tool "XYZ"
                BEGIN ToolAppData
                ORDINALITY 2:1
                END ToolAppData
        END ApplicationData
        RelationshipName “OrgCode2“
        SourceObjectIdentifier "5000"
        SourceSequenceOrder “2:2”
        TargetObjectIdentifier "6000"
        RelationshipExpression “if (stringequal %1 “AAE”) return(substring
        (%2,1,3) else if  (stringequal %1 “AAG”) return(“00”) else if (stringequal
        %1 “AAC”) return (“01”) else return (“  “);”
        RelationshipType “DERIVED”
        RelationshipOrdinality "2:1"
        RelationshipBidirectional "T"
END RELATIONSHIP
```