Data Documentation Initiative (DDI) Technical Specification

Part I:

Overview

Version 3.1

October 2009

4

5

Overview of the DDI Version 3.1 Conceptual Model

Version 8
Date: October 18, 2009
Wendy Thomas, Arofan Gregory, J Gager, I-Lin Kuo, Achim Wackerow, Chris Nelson

Table of Contents

1.0 Introduction

DDI Technical Specification Part I: Overview provides an overview and technical description of the Data Documentation Initiative (DDI) Version 3 Conceptual Model. Unlike preceding versions, the DDI standard will consist of two parts – the conceptual model, and the XML Schemas and DTDs which are derived from it. This is a common approach to the standardization of XML vocabularies, and one which provides many benefits to users: the vocabulary itself becomes more consistent and comprehensible, and the conceptual model can prove a valuable asset to developers of applications which need to support the standard, as many tools now allow for XML binding directly from a model expressed in the Universal Modeling Language (UML) or its derivatives. The conceptual model is found in DDI Technical Specification, Part III: Conceptual Model.

DDI 3 reflects a revised outlook on the intended coverage of the DDI as well as developments in XML technology. After describing this shift to a new perspective for DDI coverage and the design and structure implications, this document will provide details on the structures and mechanizations used in DDI 3. The DDI Technical Specification, Part II: User's Guide provides information on the application of DDI 3 for various uses and applications.

1.1 Metadata for the Data Life Cycle

While the original DDI took its model from the codebook, it was clear early on that many were expanding that concept to mean something much broader and perhaps more complex than a traditional hardcopy codebook. With Version 3.1, we now have the capability to document the rich complexity of social science data across its life course as reflected in the Combined Life Cycle Model [Figure 1].

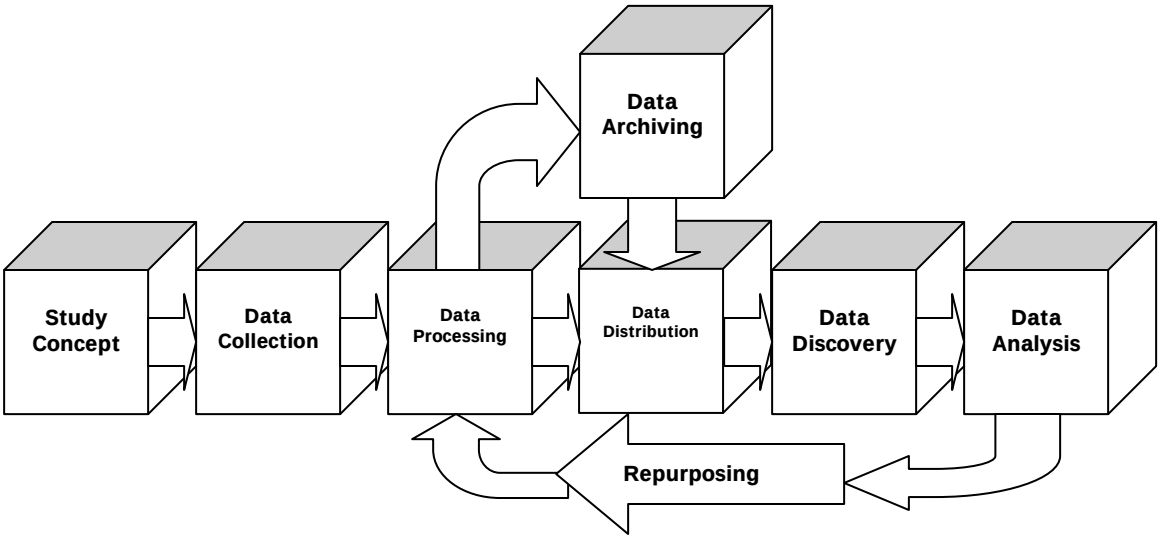


Figure 1: Combined Life Cycle Model

Historically, there has been no concept of a DDI instance existing as a study was designed, administered, and then archived. As we see in the figure above, there are now several steps to the life cycle which could be documented using DDI instances. For DDI Version 3, it is conceivable that the conceptual design of a study would be marked up in DDI, and that as the study goes through the life cycle, the DDI instance documenting it would be updated in a sequence of versions: typically, one for each stage of the life cycle.

The Combined Life Cycle Model incorporates either direct dissemination to users or dissemination through data archives and recognizes that data can be reprocessed at later points in its life cycle, creating an iterative process. This means that the life cycle is no longer linear but has become circular.
We viewed *repurposing of data* as being a secondary use of the data from a study. It was not the creation of multiple products from the same data collection such as a confidential data file, a public use file, and an aggregate data file. While multiple products could be planned for in the original conceptualization, collection, and processing of the data, *Repurposing* reflected a new conceptual framework. It may result from secondary use of a data set, or the creation of a real or virtual harmonized data set. The implications of this view include the need for defining the relationships between data products conceived of during the conception process (such as the multiple products of the United States Decennial Census) as well as the ability to define both primary and secondary data sources within the *Data Collection* phase.

The movement to a modular design for the model has been developing over time and is not a radical change in direction as much as it is recognition of the emerging consensus. It is needed to provide the flexibility for dealing with specialized data files and data sets as well as the variety of technical environments within which we currently work or are in the process of developing.

1.2 Change in Scope

DDI Version 3 represents a major change from preceding versions in another fashion: the scope has increased. Historically, DDI was focused on data archiving, and while this still remains a major focus, in Version 3 all aspects of the data life cycle will now be supported. Thus, as a data collection process proceeds, from conception to reuse, the growing set of metadata describing this activity can be collected and expressed in DDI.

This shift in scope has many repercussions in the overall design of the DDI. It means that instances will be larger to accommodate the expanded set of metadata. It also means that the simple case, where a single data file is described, no longer universally applies. Data from “studies” may be found in several files in a more flexible fashion than in preceding versions of the DDI.

These files also represent a wider range of physical data structures which need to be described.

Supporting the full life cycle means that the relationships between a study and those on which it is based may need to be recorded, and thus, groups of studies need to be described, such as a series of longitudinal studies or studies that are being compared or harmonized. A natural result of this change is providing a means of expressing comparability of studies, those which are comparable by design and those chosen for later comparison.

In addition, archives need to be able to record more information on their own activities in relation to the data. Information noting internal processing, collection management, and organizational structures required expanded support.

The metadata describing the life cycle is not complete without capturing information about the survey itself in a richer form than an image of a paper collection instrument. Many systems today allow for the re-use of questions, and thus instrument metadata are a necessary part of life cycle support.

Some other changes will be seen in the DDI Version 3 as well: optional use of a subset of HTML tagging will be supported in some of the fields where longer, human-readable text is found. Also, the handling of reusable classes, such as notes and citations has been made more uniform, increasing both the consistency of the structure and the flexibility of references to external and internal materials. The importance of other metadata standards is also recognized in this design, with the stated intent of alignment or use of several other initiatives' products.

While the changes in DDI Version 3 are ambitious in scope, one of the major design goals is to avoid making migration from Version 2.* any more arduous than necessary. The simple use of DDI for archival purposes is not radically different between versions, and mappings of all currently-used fields will be provided, as will some simple free tools for helping users.

1.3 Technology Updates

Some of the biggest changes in DDI 3 are the result of advances in XML technology. Because the use of W3C XML Schema (XSD) has become mainstream, the DDI DTD will no longer be the canonical expression of the standard. Instead, it will be a sister-product of the Schema, which – while it also describes XML instances – will express more of the validation parameters than are possible with a DTD.

The use of XML namespaces is another typical XML practice which DDI Version 3 will introduce. This allows the now-expanded vocabulary to be modularized, making it more manageable and maintainable over the long run.

It should be stated that DDI Version 3 intends to increase the degree to which the metadata it contains is sufficient to support computer processing – that is, it will go beyond being "human readable", and move toward the goal of being "machine-actionable". This is a long-term goal, and will not be taken too far in the early 3.* versions, but it is very much in keeping with the overall use of XML-based technologies now current, such as Web services.

2.0 DDI 3 Design

DDI 3 adds a lot of complexity, because it is designed to support the entire statistical lifecycle, rather than just the archival part. This places a major emphasis on being able to identify, version, and maintain the metadata throughout that process.

Further, it allows for groups of studies to be documented in relation to each other, for comparison purposes or to track versions as the metadata grows throughout the lifecycle.

Modularity supports both requirements by allowing a tighter focus on metadata that is of interest to a specific application or user. While this may seem complex, once the basic design is understood, it allows for a much more exact expression of the metadata, and, in the long term, better management and processing of that metadata.

2.1 Design Rules

The demands of the changes noted above made it clear that DDI 3 needed to outline clear design rules to ensure consistency in the creation and development of DDI 3

- Persistent sections should be separate from dynamic information. What parts change when a data file moves from one "home" to another, or changes something like its physical storage structure?
- Information modules should follow the various life cycle paths
- Information used for discovery should be in non-specialized modules
- Links should be unidirectional to avoid loops and broken links as materials are repacked or versioned
- Links should point back in time with materials later in the lifecycle pointing to existing materials rather than going back and adding new links
- All comparisons are pair wise, comparing a source with a target.
- Groups inherit down the tree unless there is a clear local override provided
- Functionality of DDI 2.1 would be preserved
- Different types of XML elements will inherit from each other in XML schemes, to simplify programmatic processing of basic types which have many different variations throughout the lifecycle.

- Metadata will be expressed in ways which support both human-readability and machine-processing.

2.2 Relationship to DDI 2.* and Earlier

All elements and attributes in 2.0 are currently represented in 3.0. Due to options for applying a small number of elements in 2.0, some hand editing or review of contents may be required to accurately migrate them to 3.0. The greatest change will be separating information currently in section 4.0 into questionnaire, logical descriptions of variables and related items, and physical storage locations. Software will be developed by DDI to facilitate this migration.

Because DDI was originally intended to support what is now termed the “simple” case, that aspect of the migration from Version 2.* to 3.0 should be more fully automatable. Thus, if you have single-document DDI instances, these should migrate in a fairly straightforward fashion to “simple” DDI Version 3 instances. In cases where DDI Version 2.* has been used to document more than a single study, the migration may become more complex, as a set of study documentation (Version 3 instances) will need to be created from the single source file.

The biggest change to DDI instances in Version 3 will be the explicit and required use of XML namespaces. It is intended that each module described below will exist in its own namespace, and these will be reflected in one of the allowed ways in the XML files themselves. Use of XML namespaces is necessary to allow DDI to use other standard structures as well as support easy maintenance of the DDI standard XML DTDs and Schemas.

XML namespaces use a prefix to identify the module from which an element description is taken. Thus, if the logical product module has its own XML namespace, it could be given the prefix “l”. A “Variable” tag would look like:

```
<l:Variable>…</l:Variable>
```

In DDI Version 2.0, there was a single, implicit namespace. Now, each module will have a namespace, and they will be made explicit.

In the “simple” case, there will be a set of modules which correspond roughly to the DDI Version 2.* sections. A detailed mapping of DDI 2.1 to 3.0 is published separately as “DDI Change Records: Mapping from DDI 2.1 to Subsequent Versions”.

Version 2	Description	Version 3
1.0	Document Description: Citation of the XML Instance / Content Citation of the Source documents	Instance / Archive

2.0	Study Description	
2.1-2.2, 2.4-2.5	Study Description, Citation, Universe, Other Materials, Note	Study Unit
2.3	Methodology	Data Collection
3.0	File Description	Physical Data Product / Physical Instance
4.0	Data Description	
4.1, 4.2, 4.4	Variable Groups, nCube Groups, nCubes	Logical Product
4.2	Variables: 1) Question 2) Location 3) Summary Statistics 4) Everything else	 1) Data Collection 2) Physical Data Product 3) Physical Instance 4) Logical Product
5.0	Other Material	Other material class of the relevant module

Notes on Version 2 Sections:

1.0 The Archive module will hold all the information specific to the archive including holdings information and file locations. The Instance or the Study Unit and their various classes (Other Materials, Notes, Universe, and Citation) will hold the remaining material.

2.0 The materials currently in the Study Description are split between the Study Unit and the Data Collection modules roughly along the lines indicated in the table.

3.0 The Physical Data Product module contains the detailed record structure information and location information while the Physical Instance module contains information on the gross file structure as well as summary and category statistics.

4.0 Most of the material in Data Description will move to the Logical Product module with the exception of the first three items listed under Variable. Question information will become part of the QuestionScheme and Instrument section of the Data Collection, Location becomes part of the Physical Data Product (similar to the current location map section), and summary statistics will move to the Physical Instance module.

2.3 Modular Design

The need to capture metadata throughout the life cycle of the data led to the decision to create a modular structure for DDI 3, allowing creators to use only those sections or modules of the DDI that were needed at the time and then adding new modules as data progressed through the life cycle. A modular approach also supports the reality of work processes in which metadata is

captured and integrated by a number of researchers and/or automated systems. A modular, “building block” approach makes creating and assembling metadata at different locations much easier. The design of the DDI Version 3 allows greater flexibility in combining various modules within a single wrapper to describe a single data file, a related group of data files, or a related group of studies. It also allows software developers or users to select which modules of information they can handle and to ignore modules outside of their capabilities.

2.3.1. Goals for Modular Design

- To organize the modules so that they accurately record information about data and the data creation process AND contain the information on structures and relationships necessary for data discovery, extraction and manipulation
- To have basic modules that will work in all technical implementations (specialized modules may not work in all technical implementations)
- To provide specialized modules for special types of data or storage formats so that all elements in the DDI are used in a consistent way
- To provide a mechanism for organizations to identify those elements they require for use or are used and understood by their software in order to provide a profile to others wishing to exchange metadata with the organization

2.4 Versioning of DDI Specifications

Beginning with version 3.0 DDI will be using the following structure to determine versioning of the specification and schemas.

MajorVersion . MinorInvalidatingVersion . MinorValidatingVersion

Major Version: Indicates a change in coverage, scope, or functionality. A major structural remodeling of the schemas would also result in a Major Version change.

Minor Invalidating Version: Indicates that the new version contains corrections for bugs or minor changes to improve functionality of current features that may result in instances created for the previous version to be invalid when parsed against the new schemas. Correction notes are provided in the updated “DDI Change Records: Mapping from DDI 2.1 to Subsequent Versions” to allow updating of existing instances to the new version. It is very possible that individual instances that did not use specific elements would not require changes.

Minor Validating Version: Indicates that the new version contains corrections for bugs or minor changes to improve functionality of current features but that DDI instances created in the previous version will still be valid under the new version.

3.0 Schemas, Schemes, and Major Reusable Classes

DDI 3 consists of 22 DDI schemas, sets of Dublin Core and XHTML schemas, 14 schemes and an extensive number of classes. All schemas are represented by .xsd files. The DDI schemas are of four types:

- *Packaging Modules*: Maintainable DDI schemas that structure metadata items rather than contain unique metadata items
- *Scheme-Based Modules*: Maintainable DDI schemas that contain maintainable schemes within their content
- *Non-Scheme-Based Modules*: Maintainable DDI schemas that contain metadata items but no schemes
- *Sub-Modules*: Only usable as a substitution for an abstract metadata class (not independently maintainable)
- *Shared Content*: Contains metadata that is used by other DDI schemas and is not maintainable

Schemes are maintainable lists of metadata elements that organize information that may be published separately and reused by a number of studies. They are the basis for resources such as question banks, concept banks, and variable banks. The construction of schemes takes into consideration their potential reuse by others. A number of proposed resource collections have been noted by DDI members including Code Schemes for standard coding items like the NAICS codes, Geographic Structure Schemes for the NHGIS geographies, as well as question and variable banks by major archives. The availability of this material in a uniformly structured format supports both reuse and mapping for comparison purposes.

Major reusable classes are those classes that are found in the schema reusable.xsd and are used extensively to structure common features like identification, reference, citations, coverage, other material, and notes. Since all of the schemas import reusable, the metadata classes found in this schema are available for use throughout the DDI instance.

3.1 XML Schemas

The schemas are listed by type. Each description includes the schema .xsd name, the abbreviation used as element name prefix in the schemas, and the official namespace for the schema. The description is followed by a list of the elements found in the root element or elements.

3.1.1 Packaging Modules

Schema Name: ***instance.xsd*** ***[none]***
Namespace: ***ddi:instance:3_1***

The DDI Instance module provides a single root element for containing all types of DDI instances. This is important because processing applications may deal with many types of XML, and they need to have a single known starting point for processing DDI XML instances.

It should be noted that DDI Instance (and DDI XML generally) is designed to be used both as a persistent format and a temporary format for transfer between applications. As a result of this, there is no assumption that a given set of metadata will be expressed in an instance the same way twice. What is versioned, maintained, and referenced in the DDI 3 is the metadata itself, rather than the XML which expresses that metadata. While this might seem like a minor distinction it has major implications for how applications are developed.

Elements contained in root element [minimum..maximum]:

r:Citation [0..1]
r:Coverage [0..1]
g:Group [0..n]
g:ResourcePackage [0..n]
g:LocalHoldingPackage [0..1]
s:StudyUnit [0..n]
r:OtherMaterial [0..n]
r:Note [0..n]
TranslationInformation [0..1]

Schema Name: ***group.xsd*** ***g***
Namespace: ***ddi:group:3_1***

This module provides the XML structure within which other modules live. This module has three top level elements. Group describes the sub-groups and study units that are part of the group as well as additional elements which provide information concerning the inheritance and sharing of metadata within the group. The basic relationship structure is provided by a set of attributes which describe the organizing principles of the specified group.

A second top-level element is Resource Package. This is used to describe maintainable modules or schemes which may be used by multiple study units outside of a group structure. The third top-level element is a Local Holding Package which allows inclusion of a depository item (Study Unit or Group) and the addition of local archive and other information without having to reversion the depository item.

Elements contained in root elements [minimum..maximum]:
Group
r:Citation [0..1]

Abstract [0..n]
Purpose [1..n]
r:SeriesStatement [0..1]
r:FundingInformation [0..n]
r:Coverage [0..1]
r:UniverseReference [0..1]
r:OtherMaterial [0..n]
a:Archive [0..1]
r:Note [0..n]
Concepts [0..n]
DataCollection [0..n]
LogicalProduct [0..n]
PhysicalDataProduct [0..n]
StudyUnit [0..n]
SubGroup [0..n]
cm:Comparison [0..1]
pr:DDIProfile [0..n]
DDIProfileReference [0..n]

ResourcePackage
r:Citation [0..1]
Abstract [0..n]
Purpose [1..n]
r:FundingInformation [0..n]
r:Coverage [0..1]
r:UniverseReference [0..1]
r:OtherMaterial [0..n]
a:Archive [0..1]
r:Note [0..n]
Concepts [0..n]
DataCollection [0..n]
LogicalProduct [0..n]
PhysicalDataProduct [0..n]
pi:PhysicalInstance [0..1]
cm:Comparison [0..1]
pr:DDIProfile [0..n]
DDIProfileReference [0..n]
a:OrganizationScheme [0..n]
c:ConceptScheme [0..n]
c:GeographicLocationScheme [0..n]
c:GeographicStructureScheme [0..n]
c:UniverseScheme [0..n]
d:ControlConstructScheme [0..n]
d:InterviewerInstructionScheme [0..n]
d:QuestionScheme [0..n]

l:CategoryScheme [0..n]
l:CodeScheme [0..n]
l:NCubeScheme [0..n]
l:VariableScheme [0..n]
p:PhysicalStructureScheme [0..n]
p:RecordLayoutScheme [0..n]

LocalHoldingPackage
g:DepositoryStudyUnitReference
g:DepositoryGroupReference
g:LocalAddedContent

Schema Name: studyunit.xsd s
Namespace: ddi:studyunit:3_1

This module contains the metadata specific to a single study unit, and as such corresponds to a DDI 2.0 instance in many ways. It should be noted that within DDI 3, the study unit can always provide local overrides to inherited metadata found in the groups and sub-groups of which it may be a part. It is always possible to express all of the metadata regarding a particular study unit as a single, simple DDI 3 instance.

Elements contained in root element [minimum..maximum]:
r:Citation [1..1]
Abstract [1..n]
r:UniverseReference [1..n]
r:SeriesStatement [0..1]
r:FundingInformation [0..n]
Purpose [1..n]
r:Coverage [0..1]
r:AnalysisUnit [0..n]
AnalysisUnitsCovered [0..n]
KindOfData [0..n]
r:OtherMaterial [0..n]
r:Note [0..n]
r:Embargo [0..n]
c:ConceptualComponent [0..n]
d:DataCollection [0..n]
l:BaseLogicalProduct [0..n]
p:PhysicalDataProduct [0..n]
pi:PhysicalInstance [0..n]
a:Archive [0..1]
pr:DDIProfile [0..n]
DDIProfileReference [0..n]

3.1.2 Scheme-Based Modules

Schema Name: ***archive.xsd*** ***a***
Namespace: ***ddi:archive:3_1***

This module provides metadata on archive specific information such as call number and local processing, LifeCycleEvents for the data or metadata, and information on all organizations or individuals associated with the contents of the instance using the OrganizationScheme. Note that for DDI an “archive” is any individual or organization that acts as the maintainer of the DDI content. In this sense it can describe the original researcher, a data production agency, a library or an archive. It can be contained directly in any of the packaging schemas listed in 3.1.1.

Elements contained in root element [minimum..maximum]:

- ArchiveModuleName [0..n]
- r:Label [0..n]
- r:Description [0..n]
- ArchiveSpecific [1..1]
- OrganizationScheme [1..1]
- r:LifecycleInformation [0..1]
- r:OtherMaterial [0..n]
- r:Note [0..n]

Schema Name: ***conceptualcomponent.xsd*** ***c***
Namespace: ***ddi:conceptualcomponent:3_1***

This module allows for the documentation of conceptual components of the metadata – which concepts are used, and how they are defined, grouped, and organized into schemes. It also contains a UniverseScheme to descript the coverage and structure of the studies universe, and two geographic schemes. GeographicStructureScheme is used to capture the top level structural types covered by the study. GeographicLocationScheme provides the specific location identifications for the structures described. It can be attached to any of the various types of DDI instance (groups, study units, resources).

Elements contained in root element [minimum..maximum]:

- ConceptualComponentModuleName [0..n]
- r:Label [0..n]
- r:Description [0..n]
- r:Coverage [0..1]
- r:OtherMaterial [0..n]
- r:Note [0..n]
- ConceptScheme [0..n]
- ConceptSchemeReference [0..n]

UniverseScheme [0..n]
UniverseSchemeReference [0..n]
GeographicStructureScheme [0..n]
GeographicStructureSchemeReference [0..n]
GeographicLocationScheme [0..n]
GeographicLocationSchemeReference [0..n]

Schema Name: datacollection.xsd d
Namespace: ddi:datacollection:3_1

This module provides for the description of the data collection process. This includes methodology, collection events, question schemes, control constructs that organize questions and text in specific orders, instruments, interviewer instructions, and processing associated with the data collection. It can be attached to any of the various types of DDI instances.

Elements contained in root element [minimum..maximum]:

DataCollectionModuleName [0..n]
r:Label [0..n]
r:Description [0..n]
r:Coverage [0..1]
r:OtherMaterial [0..n]
r:Note [0..n]
Methodology [0..1]
CollectionEvent [0..n]
QuestionScheme [0..n]
ControlConstructScheme [0..n]
InterviewerInstructionScheme [0..n]
Instrument [0..n]
ProcessingEvent [0..n]

Schema Name: logicalproduct.xsd l
Namespace: ddi:logicalproduct:3_1

This module describes the logical product of a study unit – or a shared logical product within a group or subgroup, or resource. This includes descriptions of variables, categories, category schemes, code schemes, NCubes, and information on data relationships such as logical record content, unique record identifiers and complex keys for record linking. This module is very often shared by many different DDI instances, and is available in all types of DDI instances.

Elements contained in root element [minimum..maximum]:

LogicalProductName [0..n]
r:Label [0..n]
r:Description [0..n]

r:Coverage [0..1]
DataRelationship [0..n]
r:OtherMaterial [0..n]
r:Note [0..n]
CategoryScheme [0..n]
CategorySchemeReference [0..n]
CodeScheme [0..n]
VariableScheme [0..n]
VariableSchemeReference [0..n]
NCubeScheme [0..n]

Schema Name: ***physcialdataproduct.xsd*** ***p***
Namespace: ***ddi:physicaldataproduct:3_1***

This module describes the physical layout used in a data file. Note that in DDI 3 a single data set may be spread across multiple files. Because physical data structures may be reused across many instances of a study, or even for different studies, this module may appear in most types of DDI Instances (Study Unit, Group, or ResourcePackage). This allows flexibility in managing a collection of data files and their related metadata. The physical structure scheme contains descriptions of the basic physical features of a logical record and its physical storage structure.

The record layout scheme contains the details of a record layout stored in a specific structure. A number of substitution groups for RecordLayout allow for the description of various file formats.

Elements contained in root element [minimum..maximum]:
PhysicalDataProductName [0..n]
r:Label [0..n]
r:Description [0..n]
r:OtherMaterial [0..n]
r:Note [0..n]
PhysicalStructureScheme [0..n]
RecordLayoutScheme [0..n]

3.1.3 Non-Scheme-Based Modules

Schema Name: ***comparative.xsd*** ***cm***
Namespace: ***ddi:comparative:3_1***

Comparative provides metadata about the comparison of study units with a group or sub-group, comparison to an external standard, or comparison between two or more schemes in a resource package. It describes how these study units relate

to each other in terms of their universe, concepts, questions, variables, categories and code schemes

Elements contained in root element [minimum..maximum]:

- ComparisonName [0..n]
- r:Label [0..n]
- r:Description [0..n]
- ComparisonDescription [0..n]
- ConceptMap [0..n]
- VariableMap [0..n]
- QuestionMap [0..n]
- CategoryMap [0..n]
- CodeMap [0..n]
- UniverseMap [0..n]
- r:Note [0..n]

Schema Name: ***ddiprofile.xsd*** ***pr***
Namespace: ***ddi:ddiprofile:3_1***

This module allows for DDI instances to describe which elements and attributes of the DDI they use. It is possible to declare which elements are used or not used and to change optional elements to required ones. Such profiles as DDI Core serve as the model for this module, which could not be expressed in DDI 2.0 XML. Profiles can be described in a ResourcePackage element, and re-used by reference, or can be placed in-line in Group and StudyUnit modules.

Elements contained in root element [minimum..maximum]:

- DDIProfileName [0..n]
- r:Label [0..n]
- r:Description [0..n]
- XPathVersion [1..1]
- DDINamespace [0..1]
- XMLPreixMap [1..n]
- Instructions [0..n]
- Used [0..n]
- NotUsed [0..n]

Schema Name: ***physicalinstance.xsd*** ***pi***
Namespace: ***ddi:physicalinstance:3_1***

This module describes the location and other metadata pertinent to physical instances of a data set. This module has a dependence on a physical product module, and is always specific to a particular study unit. It can contain summary statistics and category statistics directly or by referencing those held in another physical instance or by the data file represented by the physical instance.

Elements contained in root element [minimum..maximum]:

- r:Citation [0..1]
- Fingerprint [0..n]
- r:Coverage [0..1]
- r:OtherMaterial [0..n]
- r:Note [0..n]
- RecordLayoutReference [1..n]
- DataFileIdentification [1..n]
- GrossFileStructure [0..1]
- r:ProprietaryInfo [0..1]
- Statistics [0..1]

3.1.4 Sub-Modules

Schema Name: ***dataset.xsd*** ***ds***
Namespace: ***ddi:dataset:3_1***

This module is a BaseRecordLayout substitution structure. It provides a simple way of tagging data as a sub-module of a physical data product. It is best suited for non-NCube data, which can be captured in other DDI modules. The data can be grouped in a row- or column-oriented fashion, although the tag names to not reflect tabular layout, but are neutral. Data can also be entered in a random order with each item identifying its variable name and case identification.

Elements contained in root element [minimum..maximum]:

- PhysicalStructureReference [1..1]
- ArrayBase [0..1]
- Name [0..n]
- IdentifyingVariableReference [0..1]
- DefaultVariableSchemeReference [0..1]
- CHOICE: [1..1]
  - RecordSet
  - ItemSet
  - VariableSet

Schema Name: ***physicaldataproduct_ncube_inline.xsd*** ***m3***
Namespace: ***ddi:physicaldataproduct_ncube_inline:3_1***

This module is a BaseRecordLayout substitution structure. This module allows for inline descriptions of multi-dimensional data described as NCubes in the logical product.

Elements contained in root element [minimum..maximum]:

PhysicalStructureReference [1..1]
ArrayBase [0..1]
NCubeInstance [1..n]

Schema Name: physicaldataproduct_ncube_normal.xsd m1
Namespace: ddi:physicaldataproduct_ncube_normal:3_1

This module is a BaseRecordLayout substitution structure. This module contains the “normal” method of describing a multi-dimensional NCubes, placing the emphasis on the NCube as a data structure, rather than as a presentational layout.

Elements contained in root element [minimum..maximum]:

PhysicalStructureReference [1..1]
CharacterSet [1..1]
ArrayBase [1..1]
NCubeInstance [1..n]

Schema Name: physicaldataproduct_ncube_tabular.xsd m2
Namespace: ddi:physicaldataproduct_ncube_tabular:3_1

This module is a BaseRecordLayout substitution structure. This module describes the multi-dimensional data as it is presented – that is, as according to a particular tabular (2 dimensional) layout, which is especially useful when documenting historical tables of multi-dimensional data or data stored in spreadsheets.

Elements contained in root element [minimum..maximum]:

PhysicalStructureReference [1..1]
CharacterSet [1..1]
ArrayBase [1..1]
NCubeInstance [1..n]
TopLeftTableAnchor [1..1]

Schema Name: physicaldataproduct_proprietary.xsd m4
Namespace: ddi:physicaldataproduct_proprietary:3_1

This module is a BaseRecordLayout substitution structure. The module describes data held in proprietary software such as statistical software packages like SAS, SPSS, and Stata.

Elements contained in root element [minimum..maximum]:

PhysicalStructureReference [1..1]
CharacterSet [0..1]

ArrayBase [0..1]
r:Software [1..1]
DataItemAddress [0..1]
DefaultNumericDataType [0..1]
DefaultTextDataType [0..1]
DefaultDateTimeDataType [0..1]
CHOICE: [0..1]
    CodedDataAsNumeric
    CodedDataAsText
DefaultVariableSchemeReference
r:ProprietaryInfo [0..1]
DataItem [0..n]

3.1.5 Shared Content

Schema Name: ***reusable.xsd*** ***r***
Namespace: ***ddi:reusable:3_1***

This module describes XML classes which are reused in different modules throughout the DDI 3 schemas. It does not refer to reusable metadata such as that found in resource or group-based DDI instances.

Schema Name: ***dcelements.xsd*** ***dc***
Namespace: ***ddi:dcelements:3_1***

This module allows for the capture and expression of native Dublin Core elements, used either as references or as descriptions of a particular set of metadata. In DDI, the Dublin Core is not used as the primary citation mechanism – this module is included to support applications which understand the Dublin Core XML, but which do not understand DDI. This module is used wherever citations are permitted within DDI 3.

Schema Name: ***ddi-xhtml11.xsd*** ***xhtml***
ddi-xhtml11-model-1.xsd
ddi-xhtml11-modules-1.xsd
Namespace: ***http://www.w3.org/1999/xhtm***

XHTML is used in DDI 3 to allow for formatting of textual descriptions within the instance. Because of the ubiquity of XHTML and the consequent support provided for it in most development environments, it was felt that XHTML provided a better approach to formatting than a set of DDI-specific formatting tags. This module is used wherever textual descriptions which might require formatting are located within DDI 3. Only designated elements allow for XHTML tags and they are generally those that are intended to be human-readable as

opposed to machine-actionable, and whose content may require structure in order to convey the intended information. The DDI 3 schemas use the following version of the XHTML files:

XHTML Modularization 1.1
W3C Working Draft 5 July 2006
http://www.w3.org/TR/xhtml-modularization/

Schema Name: ***xml.xsd*** ***xs***
Namespace: ***http://www.w3.org/XML/1998/namespace***

This schema is used in DDI 3 to allow for use of common xml classes such as xs:lang for language formats, xs:string for string content, etc.

3.2 DDI Schemes

DDI Schemes are maintainable structures found within scheme-based XML schema. They structure information that has a high potential for being shared by a number of other study units. They can form the base of information used to populate registries such as concept or question banks and can be published as resource packages. They may be published internally to enforce consistency and comparability within an organization or project. It is also anticipated that data producers and archives may publish and share schemes that describe commonly used information like coding schemes or geographic locations for the benefit of the DDI community at large. They are listed below by their parent schema.

3.2.1 Archive

OrganizationScheme

The organization scheme within a study unit contains the identifying information on all organizations or individuals associated with the study throughout its lifecycle. It may be included in archive either in-line or by reference and multiple organization schemes can be reflected in any study. The organization scheme allows minimal identification (a name) through detailed information on relationships, roles, and contact information. At minimum the name and DDI maintenance agency ID of the maintenance agency must be declared within a published DDIInstance in order to identify the abbreviation within all internal URNs. A DDI maintenance agency registry providing both abbreviations and organization information would provide a publicly accessible reference that ensured a unique identification for individuals and organizations publishing DDI instances.

Elements contained in root element [minimum..maximum]:

OrganizationSchemeName [0..n]
r:Label [0..n]
r:Description [0..n]
Organization [0..n]
Individual [0..n]

Role [0..n]
Relation [0..n]

3.2.2 Conceptual Components

ConceptScheme

The scheme contains a list of concept terms and definitions which may be grouped into a hierarchical structure. The content can also be expressed as a complete ISO/IEC 11179 compliant data element concept structure. Within a study unit or group this contains structured concepts used by the study or studies within the DDIInstance. The concepts in the scheme are referenced by questions and variables, providing a consistent definition for all concept terms and means of locating all questions and variable used to measure or represent a single concept. Concept schemes can be published in registries to support comparability. When questions or variables from two different studies both reference a published concept the user can assume that both studies are using the same definition of the concept. This usage is common in large data collection organizations to ensure that all of their studies are using comparable concepts and definitions. Within a study, it is the combination of the universe, concept and variable representation that reflects the ISO/IEC 11179 data concept. The alternate form of an ISO/IEC 11179 data element concept was provided for use in resource packages where the link provided by the variable to the universe and representation is unavailable.

Elements contained in root element [minimum..maximum]:

ConceptSchemeName [0..n]
r:Label [0..n]
r:Description [0..n]
ConceptSchemeReference [0..n]
Vocabulary [0..1]
Concept [0..n]
DataElementConcept [0..n]
ConceptGroup [0..n]

UniverseScheme

Within a study unit or group this contains all universe statements used in the study arranged in hierarchies. A question or variable can reference one or more universes indicating that the universe of the item is the universe that satisfies both definitions. For example a “Population of the United States” universe may have one child hierarchy that divides the parent universe by gender and another child hierarchy that divides it by age. A variable linking to both “Female” and “65 years of age or older” would have a universe of “Female population of the United states who are 65 years of age or older”. By structuring these universes in a scheme, both the relationships within the universe structure and the relationships between the universes of individual questions and variables is clear. Commonly

used universe schemes could be published externally providing the same type of comparison and consistency as described for concept schemes.

Elements contained in root element [minimum..maximum]:

UniveseSchemeName [0..n]
r:Label [0..n]
r:Description [0..n]
UniverseSchemeReference [0..n]
Universe [0..n]

GeographicStructureScheme

This structure allows the contents of GeographicStructure found in coverage to be published as a resource package. GeographicStructure provides a description of the types of geographic units (countries, states, counties, places, etc.) found in a study. These structures are often the basis for linking data found in two different files or linking data to GIS systems.

Elements contained in root element [minimum..maximum]:

GeographicSchemename [0..n]
r:Label [0..n]
r:Description [0..n]
GeographicStructureSchemeReference [0..n]
r:GeographicStructure [0..n]
r:GeographicStructureReference [0..n]

GeographicLocationScheme

This structure allows the contents of GeographicLocation found in coverage to be published as a resource package. GeographicLocation provides the specific locations for the types of geographic structures described in GeographicStructure. For example, Germany, France, Canada, South Africa, Australia, and Turkey are specific locations of the GeographicStructure “country”. In addition, the individual locations may be linked to specific boundary files or describe the polygon internally using a structure similar to that found in common geographic data file metadata.

Elements contained in root element [minimum..maximum]:

GeographicLocationSchemeName [0..n]
r:Label [0..n]
r:Description [0..n]
GeographicLocationSchemeReference [0..n]
r:GeographicLocation [0..n]
r:GeographicLocationReference [0..n]

3.2.3 Data Collection

ControlConstructScheme

Control constructs are the elements that make up the flow logic of a data collection instrument. The various types include Sequence, StatementItem, QuestionConstruct, IfThenElse, RepeatUntil, RepeatWhile and Loop. As a scheme, the individual control constructs as well as master sequences can be held separately and used by a variety of instruments such as Blaise, CPSPro, CASES, and paper products.

Elements contained in root element [minimum..maximum]:

- ControlConstructSchemeName [0..n]
- r:Label [0..n]
- r:Description [0..n]
- ControlConstructSchemeReference [0..n]
- ControlConstruct [1..n]

InterviewerInstructionScheme

This scheme captures interviewer instructions in a format that can be published separately as a resource package. Interviewer instructions are listed as separate items so that they can be referenced at the appropriate place in the instrument while retaining their structure as a separate document. Interviewer instructions are frequently used for describing terminology in details as it relates to a specific data collection. They are often published as appendixes in detailed codebooks.

Elements contained in root element [minimum..maximum]:

- InterviewerInstructionSchemeName [0..n]
- r:Label [0..n]
- r:Description [0..n]
- InterviewerInstructionSchemeReference [0..n]
- Instruction [0..n]

QuestionScheme

Contains a list of questions used in the data collection instrument. This scheme can be published as a resource package or used to populate a basic question bank.

Elements contained in root element [minimum..maximum]:

- QuestionSchemeName [0..n]
- r:Label [0..n]
- r:Description [0..n]
- QuestionSchemeReference [0..n]
- QuestionItem [1..n]

3.2.4 Logical Product

CategoryScheme

A category scheme can range from all the categories used in a study to a set of specific categories that represent a single concept. To be widely usable, category

scheme construction should consider use with a specific study, considering whether questions use uncoded category schemes (the response is checked off in some manner), commonly used non-response categories, and the overall replication of categories. CategorySchemes are used directly by some questions and are organized for variables through CodeSchemes.

Elements contained in root element [minimum..maximum]:

- CategorySchemeName [0..n]
- r:Label [0..n]
- r:Description [0..n]
- CategorySchemeReference [0..n]
- CategoryGroup [0..n]
- Category [0..n]

CodeScheme

CodeSchemes apply codes to categories for use in variables or questions and can organize them into hierarchies. CodeSchemes are used by questions and variables. Variables can use a complete CodeScheme or portions of such as a specific level or range.

Elements contained in root element [minimum..maximum]:

- CodeSchemeName [0..n]
- r:Label [0..n]
- r:Description [0..n]
- CodeSchemeReference [0..n]
- CategorySchemeReference [0..1]
- HierarchyType [0..1]
- Level [0..n]
- Code [0..n]

NCubeScheme

This scheme contains a listing of NCubes in the logical product. These structures may be reused by other logical products.

Elements contained in root element [minimum..maximum]:

- NCubeSchemeName [0..n]
- r:Label [0..n]
- r:Description [0..n]
- NCube [0..n]
- NCubeGroup [0..n]
- NCubeSchemeReference [0..n]

VariableScheme

This structure contains a listing of variables used in a logical product. These variables may be reused by other logical products or as information to populate a variable bank.

Elements contained in root element [minimum..maximum]:

- VariableSchemeName [0..n]
- r:Label [0..n]
- r:Description [0..n]
- VariableSchemeReference [0..n]
- Variable [0..n]
- VariableReference [0..n]
- VariableGroup [0..n]
- VariableGroupReference [0..n]

3.2.5 Physical Data Product

PhysicalStructureScheme

This scheme contains a listing of the general physical aspects of logical records found in the study, group, or larger collection. When held as a separate scheme, this structure can be used a master listing of all records held within a collection or archive. The value of this listing is that record storage details (RecordLayout) and physical stores (Physical Instance) can be attached to the same logical record described in the Physical Structure. It is a means of identifying all records with the same intellectual (variable or NCube) content regardless of how the data is stored.

Elements contained in root element [minimum..maximum]:

- PhysicalStructureSchemeName [0..n]
- r:Label [0..n]
- r:Description [0..n]
- PhysicalStructureSchemeReference [0..n]
- PhysicalStructure [1..n]

RecordLayoutScheme

A listing of detailed record layouts linked to the Logical Record as described by any of a number of sub-module schemes or the archival record layout (ASCII fixed format or comma delimited). As with the PhysicalStructureScheme it may contain a wide range of material covering anything from a single study to the full collection of an archive.

Elements contained in root element [minimum..maximum]:

- RecordLayoutSchemeName [0..n]
- r:Label [0..n]
- r:Description [0..n]
- RecordLayoutSchemeReference [0..n]

BaseRecordLayout [1..n]

3.3 Major Reusable Classes

Reusable contains a number of complex classes that are used extensively throughout the DDI schema set. The first set of classes described below is those used to both identify and reference elements within a DDI document. The second set lists those classes that are available for use in packaging and scheme-based schemas.

3.3.1 Identification, URN and Reference

Any discussion about the interaction of the DDI 3 modules must start with the concept of identifiable, versionable, and maintainable objects. Because the various pieces of metadata making up a DDI 3 instance can be published many times in different versions throughout the lifecycle, it must be easy to find each version and understand how it fits into the development of that set of metadata.

The term “object” is used to refer to the various pieces of metadata in DDI 3. An object can be almost anything – a concept, a variable, a category, a category scheme, a question, a citation, etc. DDI 3 objects are made up of other DDI 3 objects, and there is a finite list of the different types of objects, which are termed “classes”.

At the heart of the DDI 3 design, there are classes for identifying, versioning, and maintaining an object, from which most subsequent objects inherit. Any object which can be referenced or reused must be identified uniquely. In addition to this identification, an object may also be versioned and maintained, meaning that the organization responsible for the object, as well as the version of the object can be described.

DDI 3.1 uses three forms of identification. The basic level is an *AbstractIdentifiable* which provides a urn, id, action, objectSource and UserID. An *AbstractVersionable* adds a version, version date, Version Responsibility, and Version Rationale. An *AbstractMaintainable* adds an agency, externalReferenceDefaultURI, xml:lang, and a Boolean isPublished.

An id must be unique within its Maintainable parent object. Basically, any child object is assumed to belong to the version and maintenance agency from its parent, thus the information does not have to be unnecessarily repeated. However, an object can override this inheritance – such as contents of an external reference to a maintainable object - by describing its own maintenance agency and version in its URN. Note that identifiable objects *always* belong to the same version and maintenance agency as its versionable parent. Since the id of a maintainable object is the first level of identification within an agency, the id of each maintainable object within a maintenance agency must be unique.

A good example of this is a category scheme, made up of a large set of categories. If all of the referenced categories are of the same version (say, 1.0.0) and are created and maintained by the same agency (say “us.mpc” for Minnesota Population Center) then these values are specified once for the entire category scheme, and apply as appropriate to all of its child categories. Versionable objects have an assumed version of 1.0.0 if not stated otherwise.

RULES FOR UNIQUE ID:

Within a maintenance agency:
The id of each maintainable object must be unique
Within a maintainable object:
The id of each versionable and identifiable object must be unique

3.3.1.1 Identification

All classes in the DDI schemas are identifiable or are sub-classes of identifiable complex classes. Only identifiable classes may be referenced. All classes that are identifiable are extensions of one of three types of abstract classes that describe levels of identification. All identifiable classes in DDI 3 contain a fixed attribute declaration of their identification type for ease of machine processing. Appendix 1 contains a table of all classes that are extensions of each of the identification classes.

The base abstract class is AbstractIdentifiable. All other identifiable classes build on this content. In addition to the abstract content, an AbstractIdentifiable also contains a required Boolean attribute “isIdentifiable” with the fixed content of “true”.

An AbstractIdentifiable includes the following structures:

Element:	UserID [0..n]
Attribute:	id [1..1]
	urn [0..1]
	action [0..1]
	objectSource [0..1]

The attribute id is a restricted xs:string which must start with an alphabetic character and may be followed by any alphanumeric character or any of the following non-alphanumeric characters “*”, “@”, “_”, “$”, or “-“. All identifiable classes must have an id which is unique within its maintainable object and follow the regular expression pattern “([A-Z]|[a-z]|*|@|[0-9]|_|$|\-)*”.

The urn is optional and must follow the DDI URN structure specifications. If not used, a URN can be constructed from the identification attributes provided in the identified object and its parent versionable or maintainable object.

The attribute ‘action’ is used in for inheritance situations where the identified element is being added (Add) to the inherited content, updates or overrides (Update) the inherited element, or indicates that an inherited element is not being used (Delete). Elements that ‘Update’ or ‘Delete’ an inherited element will have the SAME id as the inherited element. The attribute ‘objectSource’ allows the user to enter the DDI URN of an object that could be included by reference, but is being entered in-line in exact detail from its source. This feature supports distribution of non-published documentation with data extracts or archival versions where the problem of broken links or difficulties with resolution services must be avoided. It allows for the retention of the link for comparability purposes which providing the content in-line.

The optional element UserID allows for an identifier that is locally unique within its specific type. The required ‘type’ attribute points to the local user identification system that defines the values. If multiple UserIDs are provided they must be differentiated by the type attribute.

AbstractVersionable builds on AbstractIdentifiable and is used by classes that can be versioned to capture changes in content over time. These may be updates due to a change such as revision of the description of a category or corrections or additions to the original content. In addition to the abstract content, a VersionableID also contains a required Boolean attribute “isVersionable” with the fixed content of “true”. The following classes are added to those found in AbstractIdentifiableID.

Element:	VersionResponsibility [0..1]
	VersionRationale [0..1]
Attribute:	version [0..1]
	versionDate [0..1]

The attribute version is a restricted xs:string and must have the structure of a numeric with two extensions of a numeric separated by a period “.”. No other non-numeric characters are allowed when expressing a version number within AbstractVersionable. The regular expression for a version number is “[0-9*]+\.[0-9*]+\.[0-9*]+”The base number is a major version with extensions representing minor version changes. This structure supports easy resolution of late bound references. The value of version is assumed to be 1.0.0 if it is not stated.

The attribute versionDate is can be expressed as xs:dateTime, xs:date, xs:gYearMonth, or xs:gYear.

The two optional elements VersionResponsibility and VersionRational provide additional human-readable information for the user regarding the version change. VersionResponsibility allows specific identification of the individual or group

within a maintenance agency who made the change. This is useful in situations where multiple individuals or groups within and agency are working with the metadata and the agency wishes to track their internal processes. Because changes can only be made by the maintenance agency, this information is primarily intended for internal use. VersionRationale allows details of the rationale or purpose of the change. It is often helpful to know if the change was a spelling correction or a change in content that corrects an earlier error.

AbstractMaintainable builds on the content of AbstractVersionable by adding the identification of the maintenance agency. The maintenance agency is the agency responsible for the content and maintenance of the metadata. A DDIMaintenanceAgencyID (extention base xs:string with an attribute registryID) for the maintenance agency must be declared in the OrganizationScheme in Archive. This can be listed in-line or imported by reference. Note that when a non-maintenance agency changes DDI content, that agency becomes the new maintenance agency for the overall published instance and the maintained classes of the instance where changes took place. The only time a change can be made by a non-maintenance agency without changing the agency identification is when the change is authorized by the maintaining agency and entered at the request of that agency. The IDs of all maintainable classes within a single agency must be unique. In addition to the abstract content, a MaintainableID also contains a required Boolean attribute ‘isMaintainable’ with the fixed content of ‘true’.

Attribute: agency [0..1]

Note that agency is optional. It must be available at the DDIInstance level, but maintainable objects contained within another maintainable object inherit the agency from its parent maintainable. Note that a maintainable from another agency can be included by reference, but its use in the maintainable falls under the authority of the maintenance agency. Given the ability to publish any maintainable as a separate object, it is a good practice to provide complete identification content for maintainables within an instance. This prevents them being separated from their agency information if parsers are used that do not verify that this information is retained in the separated material.

The xml:lang attribute in AbstractMaintainable allows for declaring the default language of the content of the Maintainable object. This default can be overridden at the local level wherever an xml:lang attribute is available.

3.3.1.2 URN

The URN provides an optional means of providing complete identification for any identified item. This URN combines the id, maintenance agency, and version number into a single entity that can be used to identify any object in a non-ambiguous manner. If the URN exists and the content of the stated id disagrees

with it, the URN has priority. It would illicit an error rather than a warning in a secondary validation tool.

The DDI URN has a very specific structure. The format of this URN is the standard (DDI), the maintenance agency, the maintainable object class, id, and version number, followed by the specific object class, id, and version number (if the object is not maintainable):

:	top level FIELD separator
.	hierarchical separator within a field

Regular expression for parts of DDI URN

Field	***Regular Expression***
urn:	[Uu][Rr][Nn]
urn type:	[Dd][Dd][Ii]
agency-id (IdentifyingAgency):	[A-Za-z]+\.[A-Za-z][A-Za-z0-9\-]*
ddi-element-name:	[A-Z|a-z]+
object-id (BaseIDType):	[A-Z|a-z]+[A-Z|a-z|0-9|_|\-]*
object-version-number (NewVersionType):	([0-9]+\.[0-9]+\.[0-9]+|[0-9]+\.[0-9]+\.L|[0-9]+\.L\.L|L\.L\.L)

urn="urn:ddi:<Agency ID>:<Maintainable ddi-element-name.Maintainable object-id.Maintainable object-version-number>:<Specific ddi-element-name.Specific object-id.Specific object-version-number> "

Regular Expression for a DDI URN

[Uu][Rr][Nn]:[Dd][Dd][Ii]:[A-Za-z]+\.[A-Za-z][A-Za-z0-9\-]*:[A-Z|a-z]+\.[A-Z|a-z]+[A-Z|a-z|0-9|_|\-]*\.([0-9]+\.[0-9]+\.[0-9]+|[0-9]+\.[0-9]+\.L|[0-9]+\.L\.L|L\.L\.L)(:[A-Z|a-z]+\.[A-Z|a-z]+[A-Z|a-z|0-9|_|\-]*\.([0-9]+\.[0-9]+\.[0-9]+|[0-9]+\.[0-9]+\.L|[0-9]+\.L\.L|L\.L\.L))?

Note that the use of 'L' in the version number is only allowable when using a URN in a reference as this indicates late binding. This is explained further in Reference

Examples

URN of a maintained object

To identify of a variable scheme in DDI 3 via a URN would be as follows: **urn="urn:ddi:us.icpsr:VariableScheme.V_GENDER_SCHEME.1.0.0"**.

URN of an versionable object

All versionable objects are contained within maintainable objects. To identify of a variable in DDI 3 via a URN would be as follows:
urn="urn:ddi:us.icpsr.VariableScheme. V_GENDER_SCHEME.1.0.0: Variable .Gender.1.0.0"

URN of an identifiable object

An identifiable object may be a direct child of a maintainable object or be contained by a versionable object within a maintainable object. The full path should be provided to facilitate locating the item when referenced.

```
<DataCollection isMaintainable="true" id="DC_5698" version="2.4.0">
    <Methodology isVersionable="true" id="Meth_Type_1" version="1.0.0">
        <TimeMethod isIdentifiable="true" id="TM_1">
```

To identify the identifiable object in the above hierarchy in DDI 3 via a URN would be as follows:
urn="urn:ddi:us.icpsr:DataCollection.DC_5698.2.4.0:TimeMethod_1.1.0.0"

URN of an object that nests within its own object type

An example of this is an Individual who belongs to an Organization that is nested in another Organization. In this case each object type would be listed in order and the IDs of the full path would be provided in the URN.

```
<OrganizationScheme isMaintainable="true" id="OS_1" verson="1.0">
    <Organization isVersionable="true" id="UMICH">
        <Organization isVersionable="true" id="ICPSR">
            <Individual isVersionable="true" id="J_Doe">
```

urn="urn:ddi:us.icpsr:OrganizationScheme.OS_1.1.0.0:Individual.J_Doe.1.0.0"

3.3.1.3 Reference

All objects that contain a reference to another object have the fixed attribute isReference="true" making them easy to locate for developers. Any object that

has been identified can be referenced by another object. This theme is central to the overall structure of DDI 3. There are two major cases for the use of referencing. First to provide a relationship when two things are related, but do not have a child-parent relationship – that is, when one of them does not contain the other. This is how response domains or representations are linked to their use in a question or variable. This type of relationship also provides a needed chain of linkages from the contents of a physicalinstance back to the contents of the logical record of variables or NCubes that the data represents.

The other major case is that of reuse. If some metadata is reused in the description of many study units, or even many versions of study units, then it becomes important to be able to create a single, reusable metadata instance. This type of referencing is called 'inclusion by reference'. (This case is explored in more detail below, in the discussion of grouping and modularity.) Regardless of the reason, you need to be able to point to a specific version of any identifiable object. This is called an early-bind and creates a link to the specified version ONLY. This is generally the safest form of reference and the examples below are all early-bind situations. In some cases a late-bind option may be preferable. A reference to a ResourcePackage containing an Archive module with a collection of OtherMaterial entries might wish to obtain the most recent complete version. In this case the 'lateBound' attribute in Reference is set to "true" and the applicable version number segment(s) would be replaced with 'L' indicating the most recent available for the given segment. In other cases, such as a CategoryScheme for Industry Codes you may wish the most recent minor version of a specified major version, or '3.L.L'. In general, late-binding is only valid from right-to-left:

- 1.1.L most recent minor-minor version of 1.1
- 1.L.L most recent minor version of 1
- L.L.L most recent version

Use of late binding should be limited to situations where it is clear that version changes will not cause misunderstanding or misinformation to the user.

Whether it is a variable referencing the code scheme that provides the valid representation values or a study description referencing a previously defined collection of concepts, the mechanism for referencing is the same. An identified object is referenced either by its ID, Maintenance Agency, and version or by its structured URN. The reference can either point to an object defined within the same DDI Instance, or to an object in an external DDI Instance. If the object resides external to the DDI Instance, the isExternal attribute is set to 'true' and the URI of the DDI Instance where it is contained must be provided.

The final point to discuss in referencing is the concept of late binding. Basically, as opposed to explicitly stating the version number, one could say that the reference always refers to the latest version of an object. This is accomplished by setting the lateBound attribute on the Version element in the reference to true.

This of course assumes that the system that is processing the DDI Instance is capable of resolving such references. The attribute sourceContext allows the user to indicate the urn of the top of the most current version of a parent scheme when the immediate parent has been made part of a later version through inclusion.

All references contain the following object:

Element: Module [0..1]
Scheme [0..1]
Choice [1..2]
URN [0..1]
Sequence:
ID [0..1]
IdentifyingAgency [0..1]
Version [0..1]
Attribute: isExternal (default="false")
URI [0..1]
isReference (fixed="true")
lateBound (default="false")
sourceContext [0..1]

In its simplest form (reference to an identifiable object type within the same DDIInstance where all ID's are unique) a reference would look like the following:

<CodingInstructionsReference isReference="true" isExternal="false" lateBound="false">
<r:ID>DEV_3</r:ID>
</CodingInstructionsReference >

<CodingInstructionsReference isReference="true" isExternal="false" lateBound="false">
<r:ID>DEV_3</r:ID>
<r:URN>urn:ddi:us.icpsr:DataCollection. DataCol_1.2.0.0: Coding.DEV_3.1.0.0</r:URN>
</CodingInstructionsReference >

For a reference to a versionable object type within the same DDIInstance the reference must also include the version number unless lateBound is set to "true" indicating the most current version should be referenced.

<RelatedToReference isReference="true" isExternal="false" lateBound="false">
<r:ID>METH_2</r:ID>
<r:Version>1.1.0</r:Version>
</ RelatedToReference >

< RelatedToReference isReference="true" isExternal="false" lateBound="false">
<r:ID>METH_2</r:ID>

<r:URN>urn:ddi:us.icpsr:DataCollection. DataCol_1.2.0.0:
Methodology.METH_2.1.1.0</r:URN>
</ RelatedToReference >

Note that the added use of the URN reduces the ambiguity of the reference by providing the full path and is very valuable particularly when metadata may later be repackaged or reused. Either ID or URN or both may be used. In cases where a conflict exists between the ID and the URN, the URN takes precedence.

There is one special case for references. This is when the object being referenced has inherited metadata from its grouping structure, AND has employed local overrides (deletions, additions, or replacements). In this case the maintainable object, the module (schema) or scheme must be referenced as well. A reference is understood to be to an unmodified, inherited metadata. (Grouping is explained in the next section.)

< CodingInstructionsReference isReference="true" isExternal="false"
lateBound="false">
<r:Module isReference="true"><r:ID>DataCol_1<r:ID></r:Module>
<r:ID>DEV_3</r:ID>
<r:URN>urn:ddi:us.icpsr:DataCollection.DataCol_1.1.0.0:
Coding.DEV_3.1.0.0</r:URN>
</ CodingInstructionsReference >

< VariableReference isReference="true" isExternal="false" lateBound="false">
<r:Scheme isReference="true"><r:ID>VarScheme_1<r:ID></r:Scheme>
<r:ID>V1</r:ID>
<r:URN>urn:ddi:us.icpsr: VariableScheme.VarScheme_1.1.0.0:
Variable.V1.1.0.0</r:URN>
</ VariableReference >

Scheme references take a special construction with include both a reference to the scheme and the ability to exclude specific items from the scheme. This facilitates the reuse of schemes within a DDI instance. For example, a logical product could use this means to constrain geographic coverage by referencing the original GeographicLocationScheme and then excluding the specific GeographicLocation objects that are excluded from the coverage of the Logical Product.

3.3.2 Text Types and Dates

3.3.2.1 Text Types

DDI provides for a number of text types to support language differences, the need for structured text, and constraints on content. These basic types

Sting Type	Features

NCName	Must start with a letter and can contain alphanumeric | "_" | ":"
String	Any character string (will be read as the literal string)
InternationalString	A string with an xml:lang attribute to denote language and boolean attributes translated (default false) and translatable (default true)
StructuredString	In addition to features of InternationalString allows for XHTML structure tags in the content
IdentifiedStructuredString	Combines features of an IdentifiableID and a StructuredString
DynamicText	Structures the behavior of dynamic or static text within a question by allowing a text line to be broken into segments describing both static (literal text) and dynamic (conditional text)

The following grid shows which features are available for each type other than NCName and DynamicText. Many of the forms without ID are parts of complex elements that are identifiable.

	string	ID	xml:lang	translated	translatable	XHTML
String	X					
InternationalString	X		X	X	X	
StructuredString	X		X	X	X	X
IdentifiedStructuredString	X	X	X	X	X	X

XHTML structures supported by DDI are currently limited. Appendix 4 provides a list of valid XHTML tags and their definitions. A structured string looks like the following:

<Content xml:lang="en" translated="false" translatable="true"><xhtml:p>The text in the following paragraph is in italics**.</xhtml:p><xhtml:p><xhtml:i>**This is an example of italic typeface.**</xhtml:i></xhtml:p></Content>**

DynamicText is a specialized structure which was designed specifically to facilitate the use of dynamic text by computer assisted interviewing systems. With the increased use of CAI systems, questionnaire designers found that they could customize the textual content of a question to reflect earlier responses, such as the number of children, gender, name, etc. DDI wished to capture this information in a way that could be handled by CAI systems. Dynamic text is currently used only in the development of questions and displayed text in the control constructs used by the instrument. An example of dynamic text

<d:QuestionText xml:lang="en"><d:LiteralText><r:Text>
Since the first of

</r:Text></d:LiteralText><d:ConditionalText><d:Expression>
[MONTH]
</d:Expression></d:ConditionalText><d:LiteralText><r:Text>
2003,
</r:Text></d:LiteralText><d:ConditionalText><d:Expression>
[IF L1age<16: has anyone who lives here had their/ IF L1age>15 AND ONLY ONE PERSON 16+ IN HOUSEHOLD: have you had your/IF L1age>15 AND 2 OR MORE PEOPLE 16+ IN HOUSEHOLD: have you or anyone who lives here had their]
</d:Expression></d:ConditionalText><d:LiteralText><r:Text>
motor vehicle STOLEN OR DRIVEN AWAY WITHOUT PERMISSION, even if
</r:Text></d:LiteralText><d:ConditionalText><d:Expression>
[they/ IF L2age>15 AND ONLY ! PERSON 16+ IN HOUSEHOLD: you]
</d:Expression></d:ConditionalText><d:LiteralText><r:Text>
later got it back?
</r:Text></d:LiteralText></d:QuestionText>

If the above question was asked of someone responding with MONTH = March, L1age=23, and ONLY ONE PERSON 16+ IN HOUSEHOLD, the question would display as follows:

Since the first of March 2003, have you had your motor vehicle STOLEN OR DRIVEN AWAY WITHOUT PERMISSION, even if you later got it back?

The use of DynamicText is discussed further in Part II: Section 5.1 Question Construction. A full listing of elements and attributes using various text types is provided in Appendix 2.

3.3.2.2 Dates

All machine actionable dates in DDI 3 are expressed in standard ISO formats. The basic form of a date in DDI is the BaseDateType which is a union of ISO date types including:

xs:dateTime	yyyy-mm-ddThh:mm:ss
xs:date	yyyy-mm-dd
xs:gYearMonth	yyyy-mm
xs:gYear	Yyyy
xs:duration	PnnYnnMnnDTnnHnnMnnS

Elements of type DateType allow for both date range information and historical date options to allow capturing of legacy dates in their original formats. Any element using DateType will provide a choice between a simple date expressed in BaseDateType plus an optional historic, non-ISO format OR a date range of a start and end date (with optional historic start and end dates) plus a cycle indicator in cases where a specific iteration within a cycle needs to be designated. A calendar attribute provides the option of noting the calendar type. The description of the structure a question or variable that contains a date is found in section 3.3.4.3 Date Representation.

3.3.3 Name, Label, and Description

The elements Name, Label, and Description have been provided in a number of locations where the content may be supported in registries. The convention used for elements using r:NameType is the name of the element plus 'Name', for example VariableName. Label and Description are used in the format r:Label and r:Definition. This change from version 3.0 provides closer compliance with ISO/IEC 11179-5 Information Technology – Metadata Registries (MDR) Part 5: Naming and identification principles.

Name is an extended InternationalStringType with a Boolean attribute isPreferred (default value of 'false') that can be used to designate a preferred name. Label has been changed to an extended StructuredStringType. It continues to provide attributes for identifying location variants, valid date, type, and maximum length. It now supports language and structured content. Description has not changed and continues to be a StructuredStringType supporting language and structured content. Those maintainable containing Citation, Abstract, and Purpose did not receive the Name, Label, and Description elements as it would duplicate current more detailed content. The following table provides a list of elements with r:NameType. All have corresponding r:Label and r:Description elements.

NameType is a human understandable name (word, phrase, or mnemonic) that reflects the ISO/IEC 11170-5 naming principles. A name for an element is specified within a particular context such as an element administered within a registry. When multiple names are used the preferred name should be identified by seeing the Boolean attribute @isPreferred to 'true'. When elements are administered through a registry a Name is required. Names within an administered registry should follow the naming conventions of the registry. See ISO/IEC 11179-5 Information Technology – Metadata Registries (MDR) Part 5: naming and identification principles. ISO/IEC1179-5:2005(E).

LabelType has been redefined to clarify its purpose. LabelType is an unstructured label for the element. Label provides display content of a fully human readable display for the identification of the element. DDI does not impose any length limitations on Label. If length of Label is constrained due to use of the element in a specific application, the maximum length supported should be noted in the attribute maxLength. Label may be repeated to provide content for systems that have length constraints (e.g., some versions of the following statistical packages have character length limits: SAS 40-character, SPSS 120 characters, and Stata 80 characters).
Definition provides additional detailed information regarding the element. Note that in comparing two like types of elements, definition should be used as the basis for comparison as Name or Label may be different definitions within different contexts or registries.

The following table lists the elements of type r:NameType.

ArchiveModuleName	InstrumentName
AccessTypeName	InterviwerInstructionSchemeName
CategorySchemeName	InstructionName
CodeSchemeName	LogicalRecordName
CategoryGroupName	LogicalProductName
ConceptGroupName	MultipleQuestionItemName
ComparisonName	NCubeName
CategoryName	NCubeSchemeName
CodeMapName	NCubeGroupName
ConceptualComponentModuleName	OrganizationSchemeName
ConceptName	PhysicalDataProductName
ConceptSchemeName	PhysicalStructureSchemeName
ControlConstructSchemeName	QuestionSchemeName
DDIProfileName	QuestionItemName
DataCollectionModuleName	RecordLayoutSchemeName
DataElementConceptName	UniverseName
DataRelationshipName	UniverseSchemeName
EmbargoName	VariableSchemeName
GeographicLocationSchemeName	VariableName
GenericMapTypeMapName	VariableGroupName
GeographicStructureSchemeName	

3.3.4 Citation, Coverage, OtherMaterial, and Note

Version 2.0 of the DDI allows for the description of bibliographic citations, universe descriptions, other related materials, and notes at numerous and specific places throughout its structure. Version 3.1 has pulled these out and created uniform structures for each of these classes. The reusable classes are available in each of the modules and may be linked to any element within the module. This approach increases both the consistency of the structure and the flexibility for application of references to outside materials and internal notes. A more extensive and structured type identifier is used to assist the programmer and user in sorting through the information held in each class structure.

3.3.4.1 Citation and Coverage

In earlier versions citation covered detailed bibliographic information for the study, it's sources, and related materials. For the purposes of consistency and reuse, DDI 3 has broken down that content into three parts:

- File/Section ID: This is the equivalent of holdings information in a citation [where something is located and how it is referenced]. This level of identification is found in MaintainableIDs as well as file names and call numbers with the ArchiveSpecific information.
- Citation: This is the bibliographic citation information that doesn't change [author, title, publisher, publication place and date]
- Coverage: This is the topical, spatial, and temporal coverage of the module or item. By separating this information out, it allows for local enhancement, or the identification of items covering subsets of the overall data set [for example, a separation of an international data file into individual files for each country each with its own universe description or the separation of a hierarchical file into its component record types].

Citation contains those elements that are commonly found in a bibliographic citation. While available for all packaging and scheme-bases modules, it is generally used only for those modules which are intended to be published separately. For example, if a Study Unit had a citation and contain all other modules inline, the other modules would not have separate citations. Entries for Creator and Contributor allow for the addition of a reference to an affiliated organization. All citations include the option of providing a simple Dublin Core record in addition the selected citation items. As was true in earlier versions of DDI the only required citation object is Title.

Coverage provides topical, spatial, and temporal coverage information for the content of the module. Coverage information is allowed in all of the major modules. It is assumed to be inherited from the StudyUnit or Group descriptions and the highest level description should be inclusive of the complete contents. For example, if two study units were grouped and the first contained a temporal coverage for 2000 and the other contained a temporal coverage of 2001, the temporal coverage for the group would indicate 2000-2001. Coverage is used below the StudyUnit and Group level to constrain the coverage description. This allows the archive to create subsets of data files by time, geography, or topic and clearly indicate the coverage of each file in its respective physicalinstance.

TopicalCoverage provides for both Subject and Keyword content. These are both of CodeValueType and can contain either simple content or relate the content to controlled vocabularies or established categorizations.

TemporalCoverage is a simple series of reference dates providing the time period or periods covered by the data. Dates must be recorded in standard ISO structures, but the DDI DateType provide additional options for listing dates in alternate calendar types and in alternative layouts. Requiring the ISO format ensures interoperability with both internal processing systems and external search systems.

SpatialCoverage retains features added to 2.1 to improve interoperability with geographic search engines and expands this by providing options for detailed listing of both geographic structure types (Countries, States, Cities, etc.) and specific listing of locations for these types. The minimum level of information required by SpatialCoverage includes a TopLevelReference and a LowestLevelReference. These can be simple names such as 'Europe' for TopeLevelReference and 'Country' for LowestLevelReference. This would indicate that the overall coverage is for Europe and the lowest level of geographic detail is provided at the country level. It is strongly recommended that the object Description be included in the SpatialCoverage statement as this maps to the coverage element in Dublin Core. Note that while this element allows for XHTML structural elements, all of these will be lost when the content is translated into Dublin Core. The application of the detailed contents of SpatialCoverage will be addressed in Part II section 1.3.1.3.

3.3.4.2 Other Material

OtherMaterial provides a single common structure for describing external related materials. OtherMaterial should be entered in the module most closely related to its contents. This will help ensure retention when restructuring or repackaging occurs. OtherMaterial can be linked to any identifiable element in a DDI document. If published in a resource package it could link to any number of DDI documents.

OtherMaterial provides a Citation as described in 3.3.2.1, options for both an ExternalURLReference and External URNReference, information on the MIMEType of the document for processing purposes, the ability to link the material to any identifiable object in a DDIInstance, and a type attribute to classify the type of material described.

With DDI 3.1 OtherMaterial requires the use of an ID attribute. It has also acquired the ability to identify specific segments within the referenced object. This provides support for identifying start and stop locations for textual, audio, video, XML, and images. Each type has specific tags to support their unique structures.

TYPE	**Elements**
Textual	LineParameter (StartLine, StartOffset, EndLine, EndOffset), CharacterParameter (StartCharOffset, EndCharOffset)
Audio	AudioClipType, OtherAudioCliptype, AudioClipBegin, AudioClipEnd
Video	VideoClipType, OtherVideoCliptype, VideoClipBegin, VideoClipEnd
XML	Holds the X-Pointer expression identifying a node in the XML document
Image	Shape, Coordinates

3.3.4.3 Note

The primary change in the use of notes is that they are now grouped together in a class that is available in each module of the DDI. A Note can be attached to any identifiable element by a reference from the Note, providing a level of flexibility not available in Version 2.0. In addition, a set of types had been provided to identify specific types of commonly used notes to increase capabilities for uniform processing by software systems. It simplifies the process of adding a note which is linked to multiple elements and reduces entry time by providing a single entry. It also simplifies the option of using Note during the production process for tracking comments or review requirements as it is easy to locate, add, and remove during the life cycle. Note contains a Subject, Responsibility statement, Header, Content, type, and links to one or more identifiable elements.

With DDI 3.1 Note now has a required ID and supports language variants for its header and the ability to define a language at the Note level.

3.3.5 Representation

Representation types provide a consistent means of structuring response domains for questions and representations for variables. By using a consistent structure as a base for both class sets, DDI 3 reinforces the comparability between how data was collected and how it is represented in a dataset. This section will provide the basic structure of RepresentationType and then provide each substitution group as described in reusable. Variables use the representation substitutions found in reusable directly as substitution types for VariableRepresentation with the exception of CodeRepresentation where it allows additional specifications of the use of the CodeScheme contents. QuestionItem uses local substitution types for ResponseDomain which use their respective representation types with the addition of an optional Label and Description. Questions that require a mixture of response domain types may do so by using the StructuredMixedResponseDomain as an alternative to ResponseDomain. Each representation type described below notes the related ResponseDomain and VariableRepresentation including any details of specialized use.

All representation types provide the following optional content that help to define the classification and use of the representation content. When used as question response domains these may not be relevant, however, depending on the type of response domain the user may wish to define this content.

RecommendedDataType	This element is a CodeValueType which allows for input of a simple term or reference to an

	established controlled vocabulary list. Preferably the user should select from the W3C XML Schema Part 2 list of data types with the exception of sub-string types QNAME and NOTATION. See: http://www.w3.org/TR/2001/REC-xmlschema-2-20010502/#built-in-datatypes
GenericOutputFormat	This element is a CodeValueType which allows for input of a simple term or reference to an established controlled vocabulary list. This element provides specification for the preferred output format expressed in a generic way.
@missingValue	Provides a listing as a space delimited array of values that should be treated as missing values.
@blankIsMissingValue	A Boolean attribute that when set to 'true' indicates that a blank (no content) should be treated as a missing value.
@classificationLevel	Indicates the classification of the content as: Nominal, Ordinal, Interval, Ratio, or Continuous

3.3.4.1 Text Representation

Text representation contains three attributes, a maxLength, minLength, and regExp. The first two contain content in terms of the allowed maximum and minimum length for the content string. The third, regExp provides extensive flexibility in terms of structuring the allowed content. For example, a US ZIP code although containing only numbers is actually treated as a text string because the leading zero has meaning. A text representation for a question collecting a US 5 digit ZIP code would look as follows:

<d:TextDomain maxLength="5" minLength="5" regExp="([0-9])*"

Question: d:TextDomain
Variable: l:TextRepresentation

3.3.4 2 Numeric Representation

Numeric representation is used for describing data collected or represented as counts of the measurement such as Years of Age, Number of Children, Income, and so on. Numeric representation should not be used when the number is a code representing a category for example 0=Male and 1=Female. These are CodeRepresentations.

Numeric representation provides a set of attributes including type code (see below), scale, decimalPositions, a startValue and endValue for incremental types, and an interval to indicate increment values. It also contains NumberRange to define the Low and/or High values (indicating whether or not they are inclusive), a TopCode and BottomCode, and the ability to define

contents in terms of a regular expression. NumberRange is repeatable in the case of non-contiguous number ranges. Note that missing values should be listed in the standard Representation fields rather than as specific valid in the number range.

NumericTypeCodeType	
BigInteger	An integer of unlimited size
Integer	An integer number can hold a whole number, but no fraction. Integers may be either signed (allowing negative values) or unsigned (nonnegative values only).
Long	An integer of up to 32 bits in size corresponding to an unsigned range of 0 to 4,294,967,295 or a signed range of -2,147,483,648 to +2,147,483,647
Short	An integer of up to 16 bits in size corresponding to an unsigned range of 0 to 65,535 or a signed range of -32,768 to +32,767
Decimal	A real number (allows fractions expressed as decimals)
Float	Real numbers that may be stored in scientific notation (example: 20.0005, 99.9, -5000.12, 6.02e23)
Double	Float of up to 32 bits
Count	Ordinal number of objects in a finite set, discrete
Incremental	A value that is continuous and infinite can be interval or ratio

Question: d:NumericDomain
Variable: l:NumericRepresentation

3.3.4.3 DateTime Represenation

DateTime representation describes a wide range of date time structures and is flexible enough to handle legacy datasets which may have atypical content. The attribute ‘format’ allows for non-ISO structuring of the content, for example ‘MM/DD/YYYY’. If the format is not used the ISO format is assumed. The allowed DateTime representations include:

DateTypeCodeType (ISO 8601 usage)	
DateTime	Contains both the date and time as <date>T<time>
Date	Contains the full date from the Gregorian calender YYYY-MM-DD unless an alternative format is provided
Time	Contains the full time on a 24-hour clock system unless alternative format is provided. hh:mm:ss. Precision can be dropped resulting in hh:mm or hh. A time zone can be added <time>Z using the standard time zone designation +-hh:mm or +-hh
Year	Contains the 4 digit year YYYY
Month	Contains the 2 digit month MM

Day	Contains the 2 digit day DD
MonthDay	Contains the 2 digit month followed by the 2 digit day as MM-DD unless an alternative format is provided
YearMonth	Contains the 4 digit year followed by the 2 digit month as YYYY-MM unless an alternative format is provided
Duration	Provides a duration of time represented by one of the following formats (specific format must be declared) PnnYnnMnnDTnnHnnMnnS where n is replaced with the number of unit types for example 'P3Y6M4DT12H30M0S' defines 'a period of three years, six months, four days, twelve hours, thirty minutes, and zero seconds'. Elements may be omitted if their value is zero. T is used to separate date and time elements so that P3M is 3 months and PT3M is three minutes. Alternative format P<date>T<time> 'P0003-06-04T12:30:00'.
Timespan	This is not allowed as a date type when describing an NCube dimension as it represents two dimensions. Complex structure containing <start>/<end>, <start>/<duration>, or <duration>/<end>. Start and end can follow any of the designated datetime structures and should be declared in format. <start>/<end> example: '2007-03-01T13:00:00/2008-05-11T15:30:00' <start>/<duration> example: '2007-03-01T13:00:00/P1Y2M10DT2H30M' <duration>/<end> example 'P1Y2M10DT2H30M/2008-05-11T15:30:00' For <start>/<end> expressions, if any elment are missing from the end valude, they are assumed to be the same as for the start value including the time zone if used. For example a 2 hour meeting '2007-12-14T13:30/15:30'.

Question: d:DateTimeDomain
Variable: l:DateTimeRepresentation

3.3.4.4 Category Representation

Category Representation is used by QuestionItem when no code is provided in the instrument for the selected answer and coding instructions provide information on how the selected response is captured in the raw data. For example the following response domain:

Question:
What is your marital status?
Response Domain:

O Married

O Single, never married

O Widowed

O Divorced

To facilitate this approach, CategorySchemes must be created that contain single response sets. Because a CategoryScheme can be composed of the combined contents of other CategorySchemes, common categories such as 'Don't Know' and 'Refused to answer' can be created a single CategoryScheme and included in other Category schemes where it is used. Different data collection systems handle item checkoffs in different ways and this is left to the system to handle.

Question: CategoryDomain

3.3.4.5 Code Representation

Code representation references a specific CodeScheme used to provide the question response domain or variable representation. When used by a question the display of the full question with response domain should explicitly include the code as well as the category content. Questions use only the full code scheme referenced by r:CodeRepresentation. Variables extend r:CodeRepresentation by adding CodeSubsetInfo. This allows inclusion of only stated levels of a CodeScheme, specific codes, code ranges, or only the most detailed (discrete) codes in the scheme. Details of this use are provided in Section 4.10: Variable.

Question: d:CodeDomain
Variable: l:CodeRepresentation

3.3.4.6 Geographic Representation

This is a special response domain structured for use with the collection of geographic information based on a coordinate point. It structures the information needed to process the collected data and provides fields for overriding collection specifics when the individual case cannot be collected in the standard manner. Note that this is not used with variables because in general this information is processed to produce a variety of geographic variables of text, numeric, or code types. However, the information is required to accurately process the coordinate information as it is collected. GeographicRepresentation contains two types of information. The first set of objects provides information that is common to all cases and is related to how the geographic information is gathered.

Datum	Examples: WGS84, NAD27)
CoordinateSystem	Examples: Minnesota State Plane, UTM, Lat/Long
CoordinateZone	Example: UTM Zone 17N
@format	Examples: Decimal degrees (dd.ddddd), Decimal minutes (dd.mmmmm)
@spatialPrimitive	Examples: Point, line, polygon
CoordinateSource	Examples: GPS, address matching, map

	interpretation
ErrorCorrection	Examples: Point averaging, WAAS
Offset	
GeoreferencedObject	Examples: household, village centroid
AddressMatchType	optional, for address matched coordinates only Examples: Street segment match, zip centroid

The second set of objects structures the information that is being gathered. The object CoordinatePairs allows for one or more CoordinatePairs to be collected either individually or as an array. The remaining objects capture the required information in a case where the data cannot be collected as originally planned. For example, if an alternative offset is required, or the desired georeferenced object is unavailable, or an alternative coordinate system is used. Further information on coordinate systems for georeferencing is available from the Geographer's Craft – an online textbook from the University of Colorado. http://www.colorado.edu/geography/gcraft/notes/coordsys/coordsys_f.html

Question: d:GeographicDomain

3.3.4.7 ExternalCategoryRepresentation

This is used only by Variable when it is referencing an external category representation, with or without codes, that is NOT held in a DDI structure, for example a PDF file. It provides a reference to the external category/code scheme using xs:anyURI and a description of how the information is to be used. Note that any variable using this representation type is not machine-actionable. If a DDI structured option is available it should be used. This representation type is provided to support legacy materials that contain simple references to appendices or other external category/coding schemes. If an equivalent DDI structured content is used and the maintenance agency wishes to acknowledge the original source, the original source should be listed in OtherMaterial for the LogicalProduct with a relationship reference to the variable or variables which originally used it.

Variable: l:ExternalCategoryRepresentation

4.0 Structuring Content

4.1 Versioning

Because several organizations may be involved in the creation of a set of metadata throughout the lifecycle flow the rules for maintenance, versioning, and identification must be universal. Reference to other organization's metadata is necessary for re-use and is anticipated to become very common. Accurate references require accurate versioning of the metadata content. A maintenance agency is identified by its ID as declared in a maintained or internal organization scheme. DDI will set up a registry for DDI users to provide listing of unique IDs

for maintenance agencies. Individual or organizations who are not in the registry may declare their identification within the organization scheme of the DDI instance itself.

Maintenance agencies own the objects they maintain and only they are allowed to change or version the objects they maintain. Other organizations may reference external items in their own schemes, but may not change those items. You can make a copy which you maintain, but once you do that, you own it!

If an object changes in any way, its version must change. This may be a minor change or a major change with a major change incrementing the base number and a minor change incrementing the digits to the right of the decimal. Note that the structure for version numbers is '([0-9])+(\.([0-9])+)*'. This means any numeric of any length optionally followed by a series of numbers of any length separated by a period. Provision for multiple decimal extensions supports the expression of any level of granularity needed by the maintaining agency.

Any version change at a lower level will change the version of any containing maintainable object. Typically, objects grow and are versioned as they move through the lifecycle adding or correcting content as they develop. Note that version information is only required for published metatdata, metadata that has been packaged as a DDIInstance and intended for publication. Agencies may wish to version earlier than this to track internal metadata development. When a version is not declared it is assumed to be 1.0 by default.

Instance contains an Boolean attribute isPublished to indicate when an instance is officially published implying that it may be referenced for reuse of the content. This allows DDI creators to wrap DDI content in a DDIInstance for internal use during development of the document. Setting isPublished to 'false' indicates that the material is not stable and should NOT be referenced by those outside of the document developers.

4.2 Inclusion by Reference

DDI 3 is designed for reuse. The most common form that this will take is the inclusion by reference of standard categories, coding schemes, organization schemes, questions, variables, concepts, universes, and geographies. The value of inclusion by reference is two-fold. First, it makes the use of large commonly used structures, like ISCO categories and codes for occupations, easy to include in local metadata. Even the first version of DDI had an element for 'standard categories' which allowed pointing to an external listing of complex coding schemes for occupations, industries, and geographic locations. DDI 3 has developed this idea further, adding the ability to reference a DDI compliant structure thus making the content machine actionable as well as human-readable. DDI 3 has also expanded where the feature can be used by creating modules and schemes to house and publish these reusable pieces of metadata.

Secondly, the reuse of metadata by reference provides implicit comparability between studies. If Study A is using the 2000 version of the North American Industrial Classification Scheme (NAICS) by referencing an external publication of a DDI CategoryScheme and CodeScheme, and Study B includes the same object by reference, the user can conclude comparability between Study A and Study B for this object.

Inclusion by reference can take place at three levels: inclusion of a module, inclusion of a scheme, and inclusion of an object within a scheme. A StudyUnit may consist of a citation plus a list of references to externally published DataCollection, LogicalProduct, PhysicalDataStructure, and PhysicalInstance modules. If version copies are maintained, this provides a means of clearly identifying those sections that have been retained and those that have changed with each version.

Included objects can be modified at the local level with the use of Add, Update, and Delete as described in Section 3.3.1.1. Note that Updates to non-identified objects are made at the level of their parent identifiable. The updated identifiable should include the full content of the identified variable including those sections that do not differ from the original object found in the included object.

4.3 Controlled Vocabularies

There are many points in the DDI schemas where a controlled vocabulary is desired, but no single classification can be (or has been) identified which would be acceptable to all user communities. DDI 3 provides a CodeValueType that allows for use of a simple descriptive term while also supporting the use of an externally described controlled vocabulary. A set of fields has been made available for identifying the following information about the controlled vocabulary:

(1) The identifier/name of the controlled vocabulary
(2) The maintaining agency of the controlled vocabulary
(3) The version of the controlled vocabulary
(4) A URL where the controlled vocabulary could be found (additionally, a field could be provided for a URN)

Rather than incorporate specific controlled vocabularies in locations other than those required for interoperability, DDI is supporting the option of developing and publishing controlled vocabularies expressed in genericode. DDI is publishing a number of basic vocabularies for use with DDI. These may be used directly or incorporated into local publications of controlled vocabularies that reflect those elements that are common within the DDI community and adding those that are specific to maintenance agencies. This approach supports sharing of common coding structures as well as the publication of code schemes in formats that can be mapped for comparability.

Genericode (http://www.genericode.org) is an OASIS committee specification (CS) and is designed to define controlled vocabularies and provides support for deriving new code lists from existing code lists. This is a major feature for the intended use of genericode within the DDI community. An example of Genericode used to express a controlled vocabulary is provided in Appendix 5.

The advantages of this approach address a number of stated needs with the DDI community:

- The ability to update controlled vocabularies as needed
- Supporting existing controlled vocabularies used by individual agencies
- Improve interoperability by publishing controlled vocabularies in a common language that supports mapping between existing controlled vocabularies
- Support common vocabularies without limiting extensions for specialized use

4.4 Simple Study

The 'simple' case is intended to represent a usage of the DDI similar to what was done in early versions: to document a single study. The simple case is modular, and supports the stages of the full life cycle, but it does not involve groups of studies. The structure of the DDI design was intended to allow those who only need to document the 'simple' case to avoid having to understand or support the full complexity of DDI Version 3.1.

A simple case is a study with a single conceptual model, with a single integrated instrument of one or more parts that is administered at one or more occasions resulting in a data set with a persistent logical structure. This logical structure may be represented by one or more physical structures that are linked to each other with predefined keys. A single physical structure may be represented by one or more physical instances whose record layout matches the physical structure but may contain differing sets of records.

The key criteria are:

- Single conceptual model
- Single instrument made up of one or more parts (ex. employer survey, worker survey)
- Single logical data structure of the initial raw data (multiple data files can be created from this such as a public use microdata file or aggregate data files)

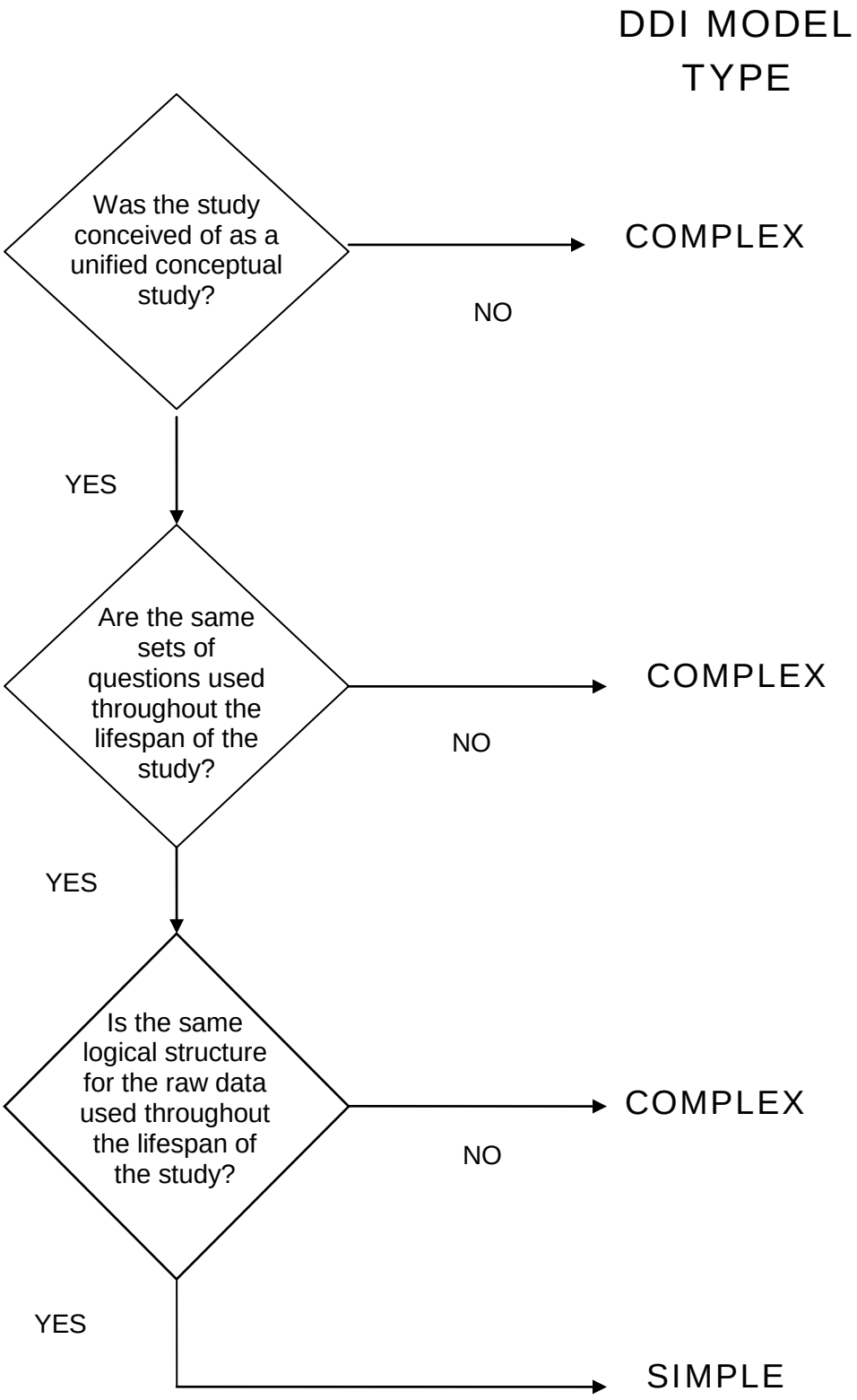
For example, the 1990 United States Census of Population and Housing can be treated as a simple study. It is based on a single coherent conceptual model, has two related questionnaires in multiple languages, and results in a raw dataset that can be defined in a single logical structure. Over 50 individual logical data

products have been created from this raw data, but all share the same data source and often many of the same variables and NCubes. The physical data files are produced in multiple formats with varying geographic coverage.

If either the instrument content (questions) or the logical data structure (variables) change over the lifetime of the study, then it becomes a complex instance requiring the use of a grouping module to define the relationships between the data sets. An example of this would be a time series like the Eurobarometer, or multiple years of a countries census where many questions are repeated over time but content changes are made to address new issues or reflect social change.

In the case that the creator of the XML does not choose to use any grouping module (if these modules are not supported by local systems), then a second XML instance must be created and any information on the relationship between the two instances will be restricted to human actionable sections of the metadata. Machine actionable relational information will be lost unless explicit comparison between the multiple studies was made.

The following flowchart illustrates the process of determining whether a given subject of documentation should be considered a ‘simple’ case or a ‘complex’ case.


DDI MODEL TYPE
Was the study conceived of as a unified conceptual study?
NO
COMPLEX
YES
Are the same sets of questions used throughout the lifespan of the study?
NO
COMPLEX
YES
Is the same logical structure for the raw data used throughout the lifespan of the study?
NO
COMPLEX
YES
SIMPLE

4.5 Group

One aspect of DDI Version 3.1 which follows from the support of the whole life cycle is the introduction of groups of studies as the subject for metadata documentation. Longitudinal studies are a good example of this. A longitudinal study is a study that is repeated at specific points in time, and thus represents a group of related studies. These need to be documented as a group in order to clearly document the repurposing of aspects of the initial study and the relationship that exists between each of the component studies in the group.

The ability to document these complex cases or groups is a major advance of DDI 3. The 'complex' case involves a series or collection of studies which are related in some way or a group of studies which are being compared. It is important to recognize which cases are 'complex' because they use features of the DDI which are potentially more difficult to understand and implement, such as group inheritance and comparison

A Group can be comprised of StudyUnits and SubGroups. A standard set of attributes describes the following dimensions for grouping:

- Time
- Instrument
- Panel
- Geography
- Datasets
- Language

A table providing the specified values and a set of decision trees for determining their value is provided in Appendix 3. Note that in all cases these attributes are providing general information on the relationships between the StudyUnits and SubGroups which comprise the Group (or SubGroup) that are intended to assist the programmer in anticipating the types of comparison or repletion patterns they will need to address. For example, if an individual StudyUnit within a group has content in three languages (labels provided in English, German, and French) this does not make Language a grouping factor. The Language attribute would be set to 'L0' 'Not a reason for grouping'. If the Group consisted of two StudyUnits say the English version of a Health Canada Survey and the French version of that survey, the Language attribute would be L2 'All original languages with full language equivalence' as Health Canada considers both versions to be original and each contains the equivalent intellectual content.

In interpreting the descriptions please note that the term 'rolling' for panel or geography means that panel waves or geographic waves were used. For example there are four panels of respondents each starting at a different point in

time and having their own repetition cycle. In panel studies this usually means a new panel wave is started each year and each panel is surveyed yearly for a limited number of years. For geography this means that there are geographic panels each consisting of say one quarter of the total Metropolitan Areas in the United States. A survey takes place yearly but the first year they survey only one geographic panel and each geographic panel is surveyed every four years. In this way the entire set of Metropolitan Areas is surveyed every four years.

4.5.1 Examples

The following section provides samples showing the grouping of studies using formal and informal Groups and a combination of both. Note that the XML structures used in these examples are for demonstration purposes only, and do not necessarily represent the actual final structure. You may wish to refer to the description of grouping properties in ”Data Documentation Initiative (DDI) Technical Specifications, Part II: High-Level Documentation, Appendix Two" for a more complete understanding of the examples given here.

4.5.1.1 Informal Group

Informal groups consist of any set of StudyUnits that the user decides to place together in a group. Informal grouping is always ‘after-the-fact’. Informal groups may be created in an academic setting to support the work of a class, identify a common source such as producer or depositor, etc. This example shows a group of StudyUnits sharing common Data Collection information - perhaps common collector – for instance, Health Canada:

```
<Group time="T0" instrument="I0" panel="P0" geography="G0" datasets="D0"
language=”L0”>
       <DataCollection>
              <CollectionEvent>CommonCollector</CollectionEvent>
       </DataCollection>
       <StudyUnit>
              <DataCollection>
                     <Instrument>INST-A</Instrument>
              </DataCollection>
              <LogicalProduct>LDP-B</LogicalProduct>
              <PhysicalDataProduct>PDP-C</PhysicalDataProduct>
              <PhysicalInstance>PDI-Y</PhysicalInstance>
       </StudyUnit>
       <StudyUnit>
              <DataCollection>
                     <Instrument>INST-B</Instrument>
              </DataCollection>
              <LogicalProduct>LDP-A</LogicalProduct>
              <PhysicalDataProduct>PDP-D</PhysicalDataProduct>
              <PhysicalInstance>PDI-X</PhysicalInstance>
       </StudyUnit>
</Group>
```

4.5.2 Formal Group

This example shows a formal group of StudyUnits sharing common properties and generally StudyUnits that form a series. For instance American Housing Survey over the course of many years:

```
<Group time="T4" instrument="I3" panel="P4" geography="G3" datasets="D2"
language="L0">
	<DataCollection>All Common Collection Info</DataCollection>
	<LogicalProduct>Common Logical Data Structure</LogicalProduct>
	<PhysicalDataProduct>Common Physical Data Product</PhysicalDataProduct>
	<StudyUnit>
		<Concept>
			<Universe>1990</Universe>
		</Concept>
		<PhysicalInstance>1990</PhysicalInstance>
	</StudyUnit>
	<StudyUnit>
		<Concept>
			<Universe>1991</Universe>
		</Concept>
		<PhysicalInstance>1991</PhysicalInstance>
	</StudyUnit>
	<StudyUnit>
		<Concept>
			<Universe>1992</Universe>
		</Concept>
		<PhysicalInstance>1992</PhysicalInstance>
	</StudyUnit>
</Group>
```

4.5.3 Nested Formal Groups

This example shows nested formal Groups, for instance, the Current Population Survey, which provides a sub set of topical questions on a monthly basis. The top level Group contains the basic set of questions, which apply to every month. The next level Group contains the topical questions for a given month:

```
<Group time="T2" instrument="I3" panel="P4" geography="G4" datasets="D4"
language="L0">
	<DataCollection>
		<ResearchInstrument>
			<Question>Question1</Question>
			<Question>Question2</Question>
			<Question>Question3</Question>
		</ResearchInstrument>
	</DataCollection>
	<SubGroup time="T2" instrument="I1" panel="P4" geography="G4"
datasets="D2" language="L0">
		<DataCollection>
			<ResearchInstrument>
				<Question>Question4</Question>
				<Question>Question5</Question>
			</ResearchInstrument>
		</DataCollection>
		<LogicalProduct>Jan Logical Data Structure</LogicalProduct>
```

```
            <PhysicalDataProduct>Jan Physical Data
Product</PhysicalDataProduct>
            <StudyUnit>
                  <Concept>
                        <Universe>Jan1999</Universe>
                  </Concept>
                  <PhysicalInstance>Jan1999</PhysicalInstance>
            </StudyUnit>
            <StudyUnit>
                  <Concept>
                        <Universe>Jan2000</Universe>
                  </Concept>
                  <PhysicalInstance>Jan2000</PhysicalInstance>
            </StudyUnit>
            <StudyUnit">
                  <Concept>
                        <Universe>Jan2001</Universe>
                  </Concept>
                  <PhysicalInstance>Jan2001</PhysicalInstance>
            </StudyUnit>
      </SubGroup>
      <SubGroup time="T2" instrument="I1" panel="P4" geography="G4"
datasets="D2" language=”L0”>
            <DataCollection>
                  <ResearchInstrument>
                        <Question>Question4</Question>
                  </ResearchInstrument>
            </DataCollection>
            <LogicalProduct>Feb Logical Data Structure</LogicalProduct>
            <PhysicalDataProduct>Feb Physical Data
Product</PhysicalDataProduct>
            <StudyUnit>
                  <Concept>
                        <Universe>Feb1999</Universe>
                  </Concept>
                  <PhysicalInstance>Feb1999</PhysicalInstance>
            </StudyUnit>
            <StudyUnit>
                  <Concept>
                        <Universe>Feb2000</Universe>
                  </Concept>
                  <PhysicalInstance>Feb2000</PhysicalInstance>
            </StudyUnit>
            <StudyUnit>
                  <Concept>
                        <Universe>Feb2001</Universe>
                  </Concept>
                  <PhysicalInstance>Feb2001</PhysicalInstance>
            </StudyUnit>
      </SubGroup>
</Group>
```

4.5.4 Mixed Groups

This example shows an informal Group containing both StudyUnits and formal Groups, for instance studies funded by United States Department of Housing and Urban Development, grouped together. This group contains one StudyUnit, and a formal Group representing the American Housing Survey:

```
<Group time="T0" instrument="I0" panel="P0" geography="G0" datasets="D0"
language=”L0”>
      <DataCollection>
            <CollectionEvent>CommonCollector</CollectionEvent>
      </DataCollection>
      <StudyUnit>
            <DataCollection>
                  <Instrument>INST-A</Instrument>
            </DataCollection>
            <LogicalProduct>LDP-B</LogicalProduct>
            <PhysicalDataProduct>PDP-C</PhysicalDataProduct>
            <PhysicalInstance>PDI-Y</PhysicalInstance>
      </StudyUnit>
      <StudyUnit>
            <DataCollection>
                  <Instrument>INST-B</Instrument>
            </DataCollection>
            <LogicalProduct>LDP-A</LogicalProduct>
            <PhysicalDataProduct>PDP-D</PhysicalDataProduct>
            <PhysicalInstance>PDI-X</PhysicalInstance>
      </StudyUnit>
      <SubGroup time="T4" instrument="I3" panel="P4" geography="G3"
      datasets="D2" language=”L0”>
            <DataCollection>Common Collection Info</DataCollection>
            <LogicalProduct>Common Logical Data Structure</LogicalProduct>
            <PhysicalDataProduct>Common Physical Data
      Product</PhysicalDataProduct>
            <StudyUnit>
                  <Concept>
                        <Universe>1990</Universe>
                  </Concept>
                  <PhysicalInstance>1990</PhysicalInstance>
            </StudyUnit>
            <StudyUnit>
                  <Concept>
                        <Universe>1991</Universe>
                  </Concept>
                  <PhysicalInstance>1991</PhysicalInstance>
            </StudyUnit>
            <StudyUnit>
                  <Concept>
                        <Universe>1992</Universe>
                  </Concept>
                  <PhysicalInstance>1992</PhysicalInstance>
                  </StudyUnit>
      </SubGroup>
</Group>
```

4.6 Resource Packages

A resource package is a means of packaging any maintainable that is not being published as part of a StudyUnit or Group. ResourcePackage structures materials for publication that are intended to be reused by multiple studies such as various schemes and modules. Note that the modules StudyUnit, Group, and PhysicalInstance cannot be published as resources packages. StudyUnit and Group are packaging structures in and of themselves and therefore do not

require ResourcePackage for publication. PhysicalIntance cannot be reused as it is the metadata for a specific external data file and its identical copies. Data that is published inline as either DataSet or PhysicalDataStructure_NCube_Inline could be published an object within a RecordLayoutScheme as both of these structures are substitutions for the RecordLayout structure that comprises the contents of a RecordLayoutScheme.

4.7 Local Holding Packages

The Local Holding Package allows publication of a local copy of deposited metadata in the format of a Study Unit without changing its maintenance agency or version, while still allowing for the addition of local processing information or value added materials. Prior to 3.1 this usage could be accommodated for deposited groups of studies but not for individual study units. Local Holding Packages contain a DepositoryStudyUnitReference andLocalAddedContent. The Local Added Content is provided in the form of a StudyUnit allowing for the provision of added content at any level of the study. It helps to clearly differentiate locally added material from original material and simplifies the incorporation of any version updates for the original study unit.

4.8 Comparison

Comparison is an area in DDI 3 that will continue to develop. Consensus was reached between the SRG and the Comparison Working Group to focus on comparison of universes, concepts, questions, categories, codes, and variables. Additional work will be required to develop comparison of various methodologies and data collection processes. Comparison in a broad sense, takes place between two or more study units as either comparison-by-design or ad-hoc-comparison. DDI 3 allows for either method.

Comparison-by-design can be encoded as inheritance from a base structure (concept, question, or variable), or through use of a more detailed item-by-item comparison structure. Ad-hoc-comparison must be done using the comparison structure. This structure provides for pair-wise comparison of individual concept, question, or variable items. Think of it as creating a harmonized structure, where each study unit is compared with the harmonized structure. Comparison between study units works on the principle "If A=B and A=C then B=C." The item level mapping structure allows the user to define the relationship, for example equivalency, parent-child, or relationship formulas.

Currently two forms of mapping are provided. The first is used for ConceptMap, VariableMap, QuestionMap, CategoryMap, and UniverseMap. It provides for identifying the source and target Schemes, a description of the correspondence, and a specific item map. Correspondence includes a human readable description of the commonality and the difference between the source and the target, a CommonalityTypeCoded that allows for use of a controlled vocabulary or a

simple string such as ‘Identical’, ‘High’, ‘Medium’, ‘Low’, or ‘None’ as well as a CommonalityWeight (0 to 1), and a UserDefinedCorrespondenceProperty consisting of a name/value pair. ItemMap provides for similar comparison for item pairs within the Source and Target Schemes.

CodeMap is slightly different in that it allows the use of d:GenerationInstruction to to define the item level correspondence. For example if the Source were a CodeScheme for marital status where the Source and Target contents were as follows:

SOURCE	TARGET
1=Never married	1=Single
2=Widowed	2=Married
3=Divorced	
4=Married	

The use of generation instruction allows for specific coding such as “IF source code < 3 THEN target code = 1” indicating that Target Code 1 is the equivalent of Source codes 1+2+3 and “IF Source code = 4 THEN Target code=2”

4.9 DDI Profile

DDI Profile is a simple collection of XPaths that describe the object within DDI that are either used or not use for particular purposes. For example CESSDA can provide a DDI profile denoting which fields it used for its online catalog and can change fields that are ‘optional’ in DDI to ‘required’ for CESSDA. Objects can be included or excluded as long as the DDI requirements are not violated. Included items can be set to a fixed or default value where appropriate or be provided with an alternate name. This structure facilitates sharing by clearly stating what is expected in the DDI metadata received or sent by an organization and defines what parts of DDI an organization or system can handle. For example software that can handle microdata structures but not NCubes.

4.10 Survey Instruments

Elements describing the questionnaire content and structure have been moved from the variable element into a sub-module of the data collection. This allows for a more coherent and richer description of the questions, their use in a survey instrument, and the means of data collection (face-to-face interview, mail out form, phone interview, CAI, etc.).

Response domains, questions, interviewer instructions ,and control constructs are defined as components of maintainable schemes so that they and their contents can be reused. This allows organizations to store and reuse questions from a question bank as well as supporting the development of larger community-wide question banks. Placing control constructs in a separate scheme allows an instrument to quickly obtain all the constructs used in an

instrument and allows multiple instrument types (Blaise, CASES, paper, etc.) to use the same control constructs and sequences without repetition.

By separating questions from the variable content and referencing them, studies that have resulted in multiple logical product creation from a single data collection process (such as Census microdata and summary statistics files) can all reference the same question description, proving a certain level of comparability between two or more logical products.

The survey instrument sections currently created for DDI 3 provide only basic minimally structured information on the development process for the questionnaire or study. Working groups have already begun to explore what is needed for adding this material at a future date.

Each instrument references the control construct containing the master sequence for the instrument content. The master sequence references other control constructs within the ControlConstructScheme that reflect routing, sequences, statement items and questions. QuestionConstructs reference QuestionItems or MultiQuestionItems housed in a QuestionScheme. Any ControlConstruct may also reference individual InterviewerInstructions found in the InterviewerInstructionScheme.

In constructing the parts of an instrument special attention should be made to separate material that is part of the use of a question within a questionnaire from that which is part of the actual question text or response content. For example, routing information is part of either an interviewer instruction or a statement within a formal flow control construct such as IfThenElse. This type of information is frequently found in print versions of questionnaires included as follows:

[IF AGE > 15] Do you have your driver's permit or license?

The part within the brackets is the routing instruction for this specific use of the question "Do you have your driver's permit or license?" In the same way, routing instructions on response categories are NOT included in the category label but are provided as routing instructions using interviewer instructions if the information needs to be visible to the interviewer. The routing itself is explicitly described by the appropriate ControlConstruct.

Q1: Gender:
Male
Female [GO TO Question 4]

'Gender' is the question text, 'Male' and 'Female' are two category labels within a CategoryScheme, Q1 is supplied in the QuestionConstruct, and 'GO TO Question 4] is an InterviewerInstruction 'If Female GO TO Question 4' and

attached to the QuestionConstruct and/or translated into the IfThenElse control construct.

4.11 Variables

Variables can be used to describe microdata data items or the dimensions and measures of NCubes. The primary differences between these two uses are as follows:

Microdata	NCube Dimension
Response domain provides the valid content for the data item as found in the data file	Response domain provides the coordinate values for the dimension that are used to identify a specific data item (cell) within the NCube matrix
Have a specified universe	Universe is assigned by the NCube
Have a specific measure	Measure is assigned by the NCube and described by a variable

Variable is primarily assembled from previously created and stored objects. This provides a certain level of comparability through references to established concepts, universe structures, questions, and embargo information. In addition to these referenced contents the Variable provides for a label and more detailed description, identification of the ResponseUnit and AnalysisUnit (using an optional controlled vocabulary and Representation information. The Variable also has three Boolean attributes so that the user can flag Variables that are temporal, geographic, or weights.

Representation allows for the description of a particular role or function of the variable in additional detail to that provided by the Variable level attributes. If the variable is a concatenation of two or more variables, ConcatenatedValue is used to list the variables used to create the current variable. References include those for weight variables, standard weights, imputation information, and coding instructions. MeasurementUnit, aggregationMeathod and additivity are listed as attributes. ValueRepresentation is expressed through one of the following substitution groups:

- TextRepresentation
- DateTimeRepresentation
- NumericRepresentaiton
- ExternalCategoryRepresentation
- CodeRepresentation

These have been described in Section 3.3.4 Representation however CodeRepresentation as used by Variable is a specialized case. CodeSchemes are constructed as simple lists, regular hierarchies, or irregular hierarchies. For hierarchies, levels are described and assigned to specific code descriptions. In addition, the most discrete levels (those with no children) are identified.

CodeRepresentation takes advantage of this information by allows the user to designate a set of included levels and included individual codes. It also allows specifying which levels will have data or if just the most discrete categories will have data. This feature is generally used to support variables used as dimensions of NCubes, but the intent is to provide for a single complete code scheme of complex variables and then to allow variables to include only those portions valid for their content. In this way a microdata variable providing a manufacturing industry code could reference the complete industrial classification and indicate which values were valid for this variable. In this way the response code for this variable retains its relationship to the larger coding scheme without the need for explicit comparison mapping.

Variables used for NCube dimensions almost always use CodeRepresentation. Dimensions require a known set of values in order to provide a coordinate address to individual cells in the NCube matrix. NCubes also use Variables to describe Measure. For example while the dimensions of an NCube can be described by the Variables Age and Sex, the content of the cells can be counts of people, counts of dogs, percentages, etc. In the case just described a Variable for each of these measures would be created (generally using a NumericRepresenation) and be referenced by the NCube as appropriate. Variables used to describe measure should have clear GenerationInstructions listed in the DataProcessing section of DataCollection.

4.12 NCubes

NCubes capture the matrix structure of aggregate data by describing the dimensions and measures expressed in the matrix through the use of Variables. NCubes are frequently the result of analyzing microdata using cross-tabulation or frequencies but can also be assembled from administrative data. The use of NCubes for description retains the relationship between values on a single dimension and between those on several dimensions. NCubes can have a single dimension or an infinite number of dimensions. Each cell within an NCube must intersect (have a value on) each dimension in one and only one point. A common NCube might be AGE by SEX by MARITAL STATUS where each dimension is described by a separate variable. When compiled into an NCube and displayed in a 2-dimensional layout it may look like the following:

Dimension rank 1: Age
Dimension rank 2: Sex
Dimension rank 3: Marital Status

		Male	Female
Under 65 years	Single	1,1,1	1,2,1
	Married	1,1,2	1,2,2
65 years	Single	2,1,1	2,2,1

and older	Married	2,1,2	2,2,2

[cells contain their coordinate location in this table]

The coordinates of each cell would be their category value in the order of the Dimension rank. So that 2,2,1 would be 65 years and older, female, single. The cell coordinate provides a link to the appropriate category label from each dimension and is later used in PhysicalDataProduct to link the storage location of the data that belongs to that cell.

An NCube, since it is a simple structure description, can contain multiple measures such as a count, percent, rank, etc. Each measure type is described by a variable. In the case of a percent or other measure requiring an independent and dependent component (numerator and denominator) the measure within the NCube can be used to identify which dimensions serve which function. This can also be described in the GenerationInstruction used by the variable describing the measure.

The cells of an NCube can also have attributes attached to them. These may be set items such as footnotes, suppression flags, source notes or set values (zero by definition). Attributes can have set values for all instances of the NCube or obtain their values from information stored in a data set (for example cell level suppression flags). Attributes can be attached to the NCube as a whole (one attribute applies to the full NCube), to each cell (separate value for each cell), or to any defined coordinate region of an NCube. For example in the above table I could define an attribute as applying only to cells that have a rank 2 value of 2 (in other words only to females).

The label, description and universe of an NCube is declared within the NCube structure. The concepts of an NCube are derived from the Variables used to describe them (Dimensions and Measures).

4.13 Data Relationship

DataRelationship defines which variables or NCubes comprise a logical record, how to identify a unique case of a specific logical record type, and how to relate two or more logical records. This section is optional only because a logical product with only category and or coding schemes used to support the response domains of a question scheme. A link to a LogicalRecord in a DataRelationship is required by all PhysicalStructure descriptions. In its simplest form a DataRelationship for a microdata file (variables) must contain the following:

```
<DataRelationship isIdentifiable=”true” id=”XX”>
        <LogicalRecord isIdentifiable=”true” id=”YY” hasLocator=”false”>
                <VariablesInRecord allVariablesInLogicalProduct=”true”/>
        </LogicalRecord>
</DataRelationship>
```

This states that all the variables in the logical product are part of a single logical record which has no variable field that identifies its record type. This is the structure used by most simple surveys. However, DataRelationship can also provide the detailed information needed to describe the content and relationship of a complex set of logical records whose contents may be described in one or more logical products. The two things that it does not do is to describe the storage order of those variables or differentiate between a single logical record stored as a single string and one stored as a series of segments. Both of these aspects are described in the PhysicalDataStructure. DataRelationship deals only with the intellectual content of a logical record and relationships between logical records.

The basic structure of DataRelationship allows for a human-readable description explaining the different record types, unique case identifiers, and record relationships. This is the section that is intended for placement within a human-readable codebook. LogicalRecord provides a description of the contents of the logical record and RecordRelationship describes pairwise relationships between needed for linking.

4.13.1 Logical Record

LogicalRecord must be provided in order to attach a data store to a logical product. It has a required Boolean attribute hasLocator. If this is set to ‘true’ VariableValueReference should be used to state the variable containing its identifying value. For example the file may contain a TYPE Variable with the value ‘H’ for a household record. It can indicate whether it contains support for multiple segments. This is generally a variable that contains a segment number. Many older files simply split records into segments, starting each segment on a new line. If you lost the record order you lost the relationship between the data in the segment and the case number. Later on variables were added that indicated the segment order. This object does not presuppose how the data is stored it simply says that the record itself contains a field that supports breaking the data into specified segments. CaseIdentification specifies the variable that allows the identification of a unique case. Normally this may be a Case number, but aggregate files can be very complex with a different set of fields required for identification depending on the value of a single field. CaseIdentification supports simple to complex instances for case identification.

The LogicalRecord must provide either VariablesInRecord for microdata files or NCubesInRecord for aggregate files. NCubesInRecord allow for identifying both NCubes and Variables to accommodate those files where case identification is provided in a variable string that is not described as part of the NCube structure. It is not used to list the variables that are used as dimensions or measures unless there is data in a file associated specifically with the variable. Both

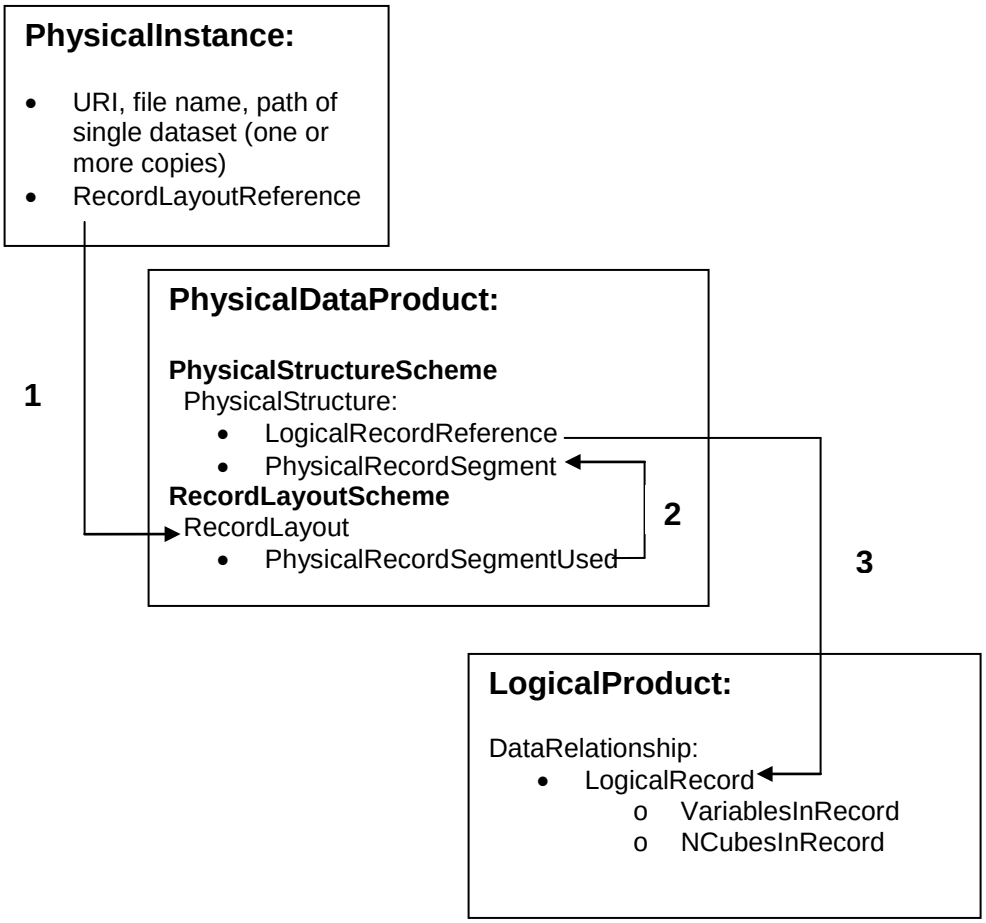
Variables and NCubes can be identified by a full scheme or schemes (allows for exclusions) and by individual variable references. If all the variables or NCubes with the logical product housing the DataRelationship are used in the logical record the Boolean attribute allVariablesInLogicalProduct or all NCubesInLogicalProduct can simply be set to 'true' and no further definition is required.

4.13.2 Record Relationship

As with all other relationship definitions RecordRelationship is pair-wise standing a Source and Target, each stating their variable location, value if appropriate and a relationship type (parent, child, or sibling). Note that this is a single variable reference for each Source and Target. If the link key is a concatenation of two or more variables, you must create a concatenated variable to use for this reference. Once the Source and Target have been identified the relationship between the Source and Target variable values can be set to Equal (default), GreaterThan, LessThan, GreaterThanOrEqual, LessThanOrEqual, or NotEqual. This simple structure and pair-wise approach provides consistent linking information for the simplest to the most complex files.

4.14 Physical Data Product and Physical Instance

The two modules PhysicalDataProduct and Physical instance provide the linkages between the logical description of a data product and one or more physical stores of the data. A single logical record can be represented by any number of physical datafiles in a wide number of physical structures (ASCII Fixed Format, ASCII Delimited, SAS, SPSS, Stata, Excel, Access, DBF, etc.). PhysicalDataProduct contains two schemes, PhysicalStructureScheme and RecordLayoutScheme. By providing these structures DDI 3 provides flexibility to archives in terms of managing their datasets and links to the related metadata. The linkage path from the externally stored dataset is as follows:



Step 1 links the datafile identified in Physical Instance with the RecordLayout described in the RecordLayoutScheme of PhysicalDataProduct. Note that when the RecordLayout has inline data (dataset of physicaldatastructure_ncube_inline), step 1 is not applicable as the data is contained within the RecordLayout.
Step 2 links the RecordLayout with the PhysicalRecordSegment of the Logical Record that is contained in the Record Layout
Step 3 links the PhysicalStructure description with the LogicalRecord in the LogicalProduct which contains the listing of variables and/or NCubes found in the LogicalRecord

4.14.1 Physical Data Product

The role of PhysicalDataProduct is to provide both a general or gross description of the physical data as well as detailed information on where data is located within a record. The PhysicalStructureScheme contains individual PhysicalStructure descriptions. A PhysicalStructure provides a link to one or more LogicalProducts, GrossRecordStructures describing one or more LogicalRecords, and optional Format and default characteristics such as decimal

positions, decimal or digit separators, data type, and missing data indicators. Note that default values are allowed at several levels providing a number of options for grouping descriptions of physical data structures. At this level the defaults should apply to the majority of dataitems. All defaults can be overridden at the data item level. Defaults were added to reduce markup for repeated content, however the use of defaults also raised a number of possibilities for how archives wished to group their data. In determining the level used for default values the user should consider which features are of greatest importance to them in processing the data and managing their collections.

The GrossRecordStructure references the LogicalRecord that describes the intellectual content of the stored data record. The LogicalRecord is described in terms of its PhysicalRecordSegements. All LogicalRecords have at least one PhysicalRecordSegment. GrossRecordStructure also provides information on the links between segments and order of appearance when the LogicalRecord has been broken into multiple physical segments. Physical segments can be stored hierarchally within a single file or in individual files by physical segment type. This aspect of storage is described in PhysicalInstance so that the same PhysicalRecordSegment can be used to describe various storage combinations as long as the content of the physical record is consistent.

RecordLayoutScheme contains one or more RecordLayouts of any substitution for BaseRecordLayout. The use of substitution structures held in separate XML schemas makes expansion to new storage types easy to incorporate and allows each substitute RecordLayout to address the specific needs and specifications of the storage type it describes. The RecordLayouts available for DDI 3 include:

- RecordLayout: The standard archival format of ASCII fixed or delimited layouts similar to those used in earlier versions of DDI. This RecordLayout is the only one located internal to the PhysicalDataProduct.
- DataSet: Inline data structure for use with microdata.
- NCube RecordLayout: The standard ASCII fixed or delimited layout similar to that used in DDI 2.0
- Inline NCube RecordLayout: Inline data structure for use with aggregate data.
- Tabular NCube RecordLayout: For aggregate data stored in a 2-dimentional tabular structure like a spreadsheet or print-like layout.
- Proprietary RecordLayout (BETA): A beta version of a generic proprietary record layout for use with common statistical packages.

The common features of all RecordLayout substitution groups are a declaration of the character set used and the array base (0|1), and a reference to the PhysicalRecordSegment contained in the Record. Each RecordLayout provides varying information specific to its type plus a listing of DataItems providing their link to the variable or NCube coordinates and the physical location of the data in the stored record or the data value (inline data). The physical location of the data

may be stated as a StartPosition with and EndPosition and/or Width, information on how to address the data item (Variable Name), or column/row combination.

4.14.2 Physical Instance

Physical Instance was designed to serve as a one-to-one relationship to a physical data file. This has been expanded slightly to allow the same physical instance to link to multiple copies of the same data file. Physical Instance contains the file name and path information for the data file, optional fingerprint for the data file, information on coverage of the data file if constrained from the overall coverage of the study, and basic file dimensions to assist in validating the content of the data file. Coverage can be constrained by creating subsets of the complete record set that would make up a full file. For example, a microdata set may be constrained to Household records only, or just records for Females. Aggregate data sets are often broken into separate files based on geographic location (all the records for a specified country or state), or by geographic structure (all county level records or all place records). The only difference between this individual files is the coverage of their records. All other features in terms of record content and layout would be the same. The gross file structure information is optional but very useful in providing users with check sums for the number of cases and number of records. Other information such as processing checks, place of production, creation software, and processing status is useful in tracking the processing of individual data files.

Physical Instance also contains summary and category level statistics. These were placed in Physical Instance because their values change with the coverage of the data file. Statistics in one physical instance can be referenced by another so that if you had four different storage formats only one of them would need to store the statistics, the other three could reference them. In addition, if statistics are held in a separately described data file, the physical instance of this file can be referenced. The user would follow the previously described link path to the logical description of the files contents. Note that category statistics can contain a single filter element so that studies such as Eurobarometer which cover multiple countries can report category level statistics broken down by country.

4.15 Extending DDI Schemas

The following approach has been proposed by the SRG as an extension methodology for use with the DDI schemas. The intent of this proposal is to make schema extensions predictable and backward-compatible, in line with type-aware XML processing and the general object-oriented features of W3C XML schema.

To extend a DDI schema, the extending agency would declare their own XML namespace, and would use the xs:import element to import the DDI schema module to be extended. Additionally, the DDI instance module would need to be imported, and a top-level extension of DDIInstance created, to hold the extended document.

The extensions would be made by declaring a type which extends a native type in the imported DDI namespace using the xs:extension or xs:restriction elements.

Elements of the extended type in the customized namespace would be declared to be of a substitution group based on an abstract, globally-declared element which corresponds to the extended or restricted DDI native type declaration. Note that this would require every extensible type in the DDI to have a global, abstract element declared for it. This modification has not been made to the current schema draft, but allows control of which parts of the DDI would be subject to extension, and which would not.

This approach is similar to that found in some other standards, and has the benefit of allowing applications which must process extended DDI documents to be able to identify and ignore the extensions, while being confident that none of the expected elements are missing.

5.0 Relationship to Other Standards

In constructing DDI special care was taken to review related standards as well as previous versions of DDI in order to provide clear mapping to the contents of outside standards or to incorporate content where appropriate. Over 25 standards were evaluated. DDI 3 currently has mapped relationships to the following standards:

- DDI 2.1 and earlier versions
- Dublin Core (Basic Bibliographic Information)
- MARC (Bibliographic Information)
- ISO/IES 11179 Data Registry
- ISO 19118 (Geography)
- SDMX (Aggregate data)
- METS (Content Wrapper)
- PREMIS (Preservation)

5.1 DDI 2.1 and Earlier

After conceptualizing the lifecycle model and the design rules for reuse, DDI 2.1 content was distributed to the schemas comprising DDI 3. Specific mapping of objects from DDI 2.1 to DDI 3 brought to light a number of specific issues which were then addressed during DDI 3 revisions. While specific objects may not always have a specific 1:1 correlation in 3.0, the content of all 2.1 objects has been captured, often in greater detail or a more consistent manner than in earlier DDI versions. DDI 3's commitment to reuse and machine-actionability resulted in creating common structures for Notes and the various forms of reference to external materials and in providing additional structure to content required to drive software and programming systems. In addition, a number of objects in earlier DDI versions were applied in specialized ways be various archives to

provide greater detail or controlled content to meet the needs of the archive. This has resulted in a number of cases where content may map to one of several places dependent upon its intended use. A full mapping of DDI 2.1 to DDI 3 is provided in Appendix 4.

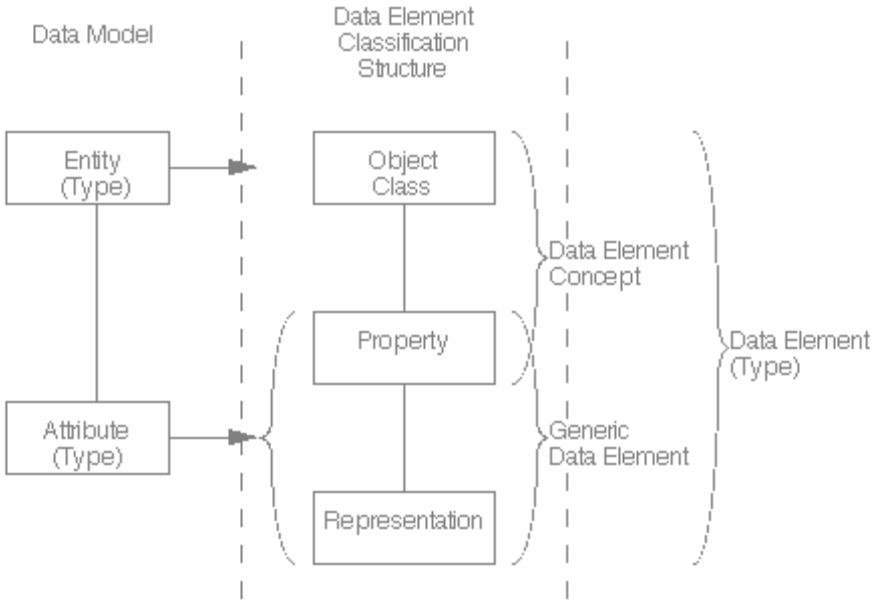
5.2 Dublin Core and MARC

All citations in DDI 3 provide the option of entering a supplemental citation in native unqualified Dublin Core. In addition, the contents of both qualified Dublin Core and the more extensive MARC record can be mapped to objects in DDI 3. DDI 3 has divided the contents of these records to a number of complex element groups within DDI to facilitate reuse of specific sub-structures. The major divisions include:

- Citation – This structure is used for both DDI content citations and citations for external materials.
- Coverage – Temporal, Topical, and Spatial coverages map to content coverage dates, subject and keyword topics, and geographic coverage elements. They are held separately in DDI 3 in order to allow coverage constraint for modules within a single StudyUnit or Group.
- Location Specific Information – Some information such as acquisition date, call number, local identifiers, etc. are related to a specific holding and are therefore located in the ArchiveSpecific section of the ArchiveModule. This facilitates packaging for transfer and incorporation into a different archive.

5.3 ISO/IEC 11179

This standard describes the structure and content of a data element as the basic building block of information. DDI 3 is particularly concerned with providing the information needed to populate an ISO/IEC 11179 data element and support a registry structure. The following diagram provides the Data Element Structure.



International Standard ISO/IEC 11179-1: Information technology – Specification and standardization of data elements – Part 1: Framework for the specification and standardization of data elements Technologies de l'informatin – Spécifiction et normalization des elements de données – Partie 1: Cadre pout la specification et la normalization des elements de données. First edition 1999-12-01 (p26) http://metadata-standards.org/11179-1/ISO-IEC_11179-1_1999_IS_E.pdf

In DDI terminology, the Object Class is defined by the universe, its Property is the concept, and the Representation is the Representation content used by the Variable that measures it. ConceptualComponent contains Universe and Concept definitions while Representation is described within the Variable. In most DDI instances it is the Variable that ties the three sections of this definition together. Note that if the Variable does not include a concept reference the instance is not compliant with ISO/IEC 11179. In addition to this means of relating the sections of a Data Element, DDI added a DataElementConcept to ConceptualComponent for the purpose of supporting an external registry through its scheme structure. DataElementConcept allows a for a description of a concept that acts as a Representation (such as Male) and provides a link to the Concept that defines the Characteristic (Sex) and the link to the Universe (Persons) thereby completing the relationship pattern without the use of Variable. In this way the schemes can be published outside of the context of a specific study and used to populate a registry of data elements.

5.4 ISO 19118 - Geography

The construction of geographic objects within DDI was done using the US FGDC which is ISO 19118 compliant. The content of the following objects map to these geographic standards:

In SpatialCoverage:

- Bounding Box
- Spatial Object (SpatialPrimative)
- TopLevelReference
- LowestLevelReference
- BoundingPolygon
- Point

In GeographicResponseDomain:

- Datum
- CoordianteSystem
- CoordinateZone
- ErrorCorrection
- Offset
- GeoreferencedObject
- CoordinatePairs
- SpatialPrimitive

The use of these fields provides search information for coordinate based search systems and detailed information needed by the geographer to determine the usefulness of a specific data set for geographic analysis.

5.5 SDMX

Careful comparison was made between DDI 2.1 nCubes and SDMX structures. In evaluating the structure and application of these two specifications it was concluded that while basic SDMX structures could be described as nCubes, not all nCubes could be described in SDMX. SDMX deals with well structured, well defined data which contains a time dimension. Not all legacy data contains well structured and well defined aggregate data and nCubes provide support for these structures. SDMX contained a more flexible approach to attaching information to regions of cells within the matrix and used a standard attribute structure to define all aspects of the matrix from the label to the cell content. SDMX requires the data cell content to be within the structure while DDI nCubes allow for the separation of metadata description and data content.

In DDI 3 the NCube structure retains the specified objects for Label, Universe, Dimensions, and Measure but adds the Attribute object and the ability to define regions of the matrix and to attach attributes to these regions. DDI 3 NCubes were designed to map to both earlier nCube structures and to SDMX providing support for using SDMX as a data transfer or storage structure.

5.6 METS and PREMIS

METS is a standard developed as an initiative of the Digital Library Federation and provides a consistent outer wrapper for digital objects described by a variety of METS profiles. The METS structure was consulted in developing the structure for the Collection and Item objects in Archive and the intent is to write and register a METS Profile for DDI.

PREMIS was brought to our attention recently and a preliminary mapping of DDI 3 to PREMIS objects was created. The focus of PREMIS is preservation and there are several elements where DDI 3 does not provide controlled content. However, with the ability to publish controlled vocabularies external to the DDI specification, we should be able to address all but a few of the PREMIS objects.

Appendix 1: URL Paths for all identified objects

(I) Identifiable
(V) Versionable
(M) Maintainable
URN identifiers start with the Maintainable Parent and works back right to left. Identified objects are listed in alphabetic order. Only the MaintainbleObjectType and the ObjectType of the object being referenced or identified is listed to the left of the equal sign. Note that if the object is identifiable and intervening parent (between object and its maintainable) is versionable, this will contain the version number that is inherited by the identifiable object
URN example for Coding
urn:ddi:us.mpc:DataCollection.DC_1.3.0.0:Coding.Code_5.1.0.0

OBJECT	PARENT (level1)	PARENT (level2)	PARENT (level 3)	NOTES
Abstract(I)	Group(M)			
Abstract(I)	StudyUnit(M)			
Access(I)	Archive(M)			
ActionToMinimizeLoss(I)	CollectionEvent(I)	DataCollection(M)		
Attribute(I)	NCube(V)	NCubeScheme(M)		
Category(V)	CategoryGroup(V)	CategoryScheme(M)		
Category(V)	CategoryScheme(M)			
CategoryGroup(V)	CategoryScheme(M)			
CategoryMap(V)	Comparative(M)			
CodeMap(V)	Comparative(M)			
Coding(I)	ProcessingEvent(I)	DataCollection(M)		
CollectionEvent(I)	DataCollection(M)			
CollectionSitutation(I)	CollectionEvent(I)	DataCollection(M)		
ComputationItem(V)	ControlConstructScheme(M)			

Concept(V)	ConceptScheme(M)			
ConceptGroup(V)	ConceptScheme(M)			
ConceptMap(V)	Comparative(M)			
CoordinateGroup(I)	NCube(V)	NCubeScheme(M)		
DataCollectionMethodology(I)	Methodology(V)	DataCollection(M)		
DataElementConcept(V)	ConceptScheme(M)			
DataFileIdentifiation(I)	PhysicalInstance(M)			
DataRelationship(I)	LogicalProduct(M)			
DefaultAccess(I)	Archive(M)			
DeviationFromSampleDesign(I)	Methodology(V)	DataCollection(M)		
Embargo(I)	StudyUnit(M)			
GeographicCoverage(I)	ConceptualComponent(M)			
GeographicCoverage(I)	DDIInstance(M)			
GeographicCoverage(I)	DataCollection(M)			
GeographicCoverage(I)	Group(M)			
GeographicCoverage(I)	LogicalProduct(M)			
GeographicCoverage(I)	PhysicalInstance(M)			
GeographicCoverage(I)	ResourcePackage(M)			
GeographicCoverage(I)	StudyUnit(M)			
GeographicCoverage(I)	SubGroup(V)	Group(M)		
Geography(I)	GeograpicStructure(V)	GeographicStructureScheme(M)		
GeograpicLocation(V)	GeographicLocationScheme(M)			
GeograpicStructure(V)	GeographicStructureScheme(M)			
GrossFileStructure(I)	PhysicalInstance(M)			
GrossRecordStructure(I)	PhysicalStructure(V)	PhysicalStructureScheme(M)		
IfThenElse(V)	ControlConstructScheme(M)			

Indivdual(V)	OrganizationScheme(M)			Individual may be nested in another individual or within an organization
Instruction(V)	InterviewerInstructionScheme(M)			
LifeCycleEvent(I)	Archive(M)			
Location(I)	Indivdual(V)	OrganizationScheme(M)		Individual may be nested in another individual or within an organization
Location(I)	Organization(V)	OrganizationScheme(M)		Organization may be nested in another organization or within an individual
LogicalRecordtype(I)	DataRelationship(I)	LogicalProduct(M)		
Loop(V)	ControlConstructScheme(M)			
Measure(I)	NCube(V)	NCubeScheme(M)		
Methodology(V)	DataCollection(M)			
ModeOfCollection(I)	CollectionEvent(I)	DataCollection(M)		
MultipleQuestionItem(V)	QuestionScheme(M)			May be nested in one or more MultipleQuestionItem(V)
NCube(V)	NCubeSchme(M)			
NCubeGroup(V)	NCubeSchme(M)			
NCubeInstance(V)	RecordLayout(V)	RecordLayoutScheme(M)		
Organization(V)	OrganizationScheme(M)			Organization may be nested in another organization or within an individual
PhysicalRecordSegment(I)	GrossRecordStructure(I)	PhysicalStructure(V)	PhysicalStructureScheme(M)	
PhysicalStructure(V)	PhysicalStructureScheme(M)			

ProcessingEvent(I)	DataCollection(M)			
ProprietaryRecordLayout(V)	RecordLayoutScheme(M)			
Purpose(I)	Group(M)			
Purpose(I)	StudyUnit(M)			
QuestionConstruct(V)	ControlConstructScheme(M)			
QuestionItem(V)	QuestionScheme(M)			May be nested in a MulitpleQuestionItem(V)
QuestionMap(V)	Comparative(M)			
RecordLayout(V)	RecordLayoutScheme(M)			
RecordRelationship(I)	DataRelationship(I)	LogicalProduct(M)		
RepeatUntil(V)	ControlConstructScheme(M)			
RepeatWhile(V)	ControlConstructScheme(M)			
Role(I)	OrganizationScheme(M)			
SamplingProcedure(I)	Methodology(V)	DataCollection(M)		
Sequence(V)	ControlConstructScheme(M)			
StatementItem(V)	ControlConstructScheme(M)			
SubGroup(V)	Group(M)			
TemporalCoverage(I)	ConceptualComponent(M)			
TemporalCoverage(I)	DDIInstance(M)			
TemporalCoverage(I)	DataCollection(M)			
TemporalCoverage(I)	Group(M)			
TemporalCoverage(I)	LogicalProduct(M)			
TemporalCoverage(I)	PhysicalInstance(M)			
TemporalCoverage(I)	ResourcePackage(M)			
TemporalCoverage(I)	StudyUnit(M)			
TemporalCoverage(I)	SubGroup(V)	Group(M)		
TimeMethod(I)	Methodology(V)	DataCollection(M)		

TopicalCoverage(I)	ConceptualComponent(M)			
TopicalCoverage(I)	DDIInstance(M)			
TopicalCoverage(I)	DataCollection(M)			
TopicalCoverage(I)	Group(M)			
TopicalCoverage(I)	LogicalProduct(M)			
TopicalCoverage(I)	PhysicalInstance(M)			
TopicalCoverage(I)	ResourcePackage(M)			
TopicalCoverage(I)	StudyUnit(M)			
TopicalCoverage(I)	SubGroup(V)	Group(M)		
Universe(V)	UniverseScheme(M)			
Variable(V)	VariableSchme(M)			
VariableGroup(V)	VariableSchme(M)			
VariableMap(V)	Comparative(M)			
VariableSet(I)	Coding(I)	ProcessingEvent(I)	DataCollection(M)	
Weighting(I)	ProcessingEvent(I)	DataCollection(M)		

Appendix 2: Special Text Type Locations

Dynamic Text Type	
Data Collection	DisplayText
Data Collection	QuestionText
Data Collection	ResponseText
Data Collection	Instruction
Identified Structured String Type	
Data Collection	DataCollectionMethodology
Data Collection	TimeMethod
Data Collection	SamplingProcedure
Data Collection	DeviationFromSampleDesign
Data Collection	ModeOfCollection
Data Collection	CollectionSituation
Data Collection	ActionToMinimizeLosses
Data Collection	Weighting
Group	Abstract
Group	Purpose
Study Unit	Abstract
Study Unit	Purpose
International String Type	
Archive	ClassDescription
Archive	LocationInArchive
Archive	Format
Archive	Media
Archive	Statement
Archive	Nickname
Archive	Keyword
Archive	Name
Archive	OrganizationName
Conceptual Component	Title
Conceptual Component	Abbreviation
Conceptual Component	Keyword
Data Collection	Label
Data Collection	ResponseUnit
Data Set	Name
DDIProfile	AlternateName
Logical Product	Purpose
Reusable	RelationshipDescription
Reusable	VersionRationale

Reusable	Name
Reusable	Title
Reusable	SubTitle
Reusable	AlternateTitle
Reusable	Publisher
Reusable	Copyright
Reusable	CoverageLimitation
Reusable	GeographyName
Reusable	RelationshipDescripton
Study Unit	AnalysisUnitsCovered
NCName	
Comparative	SourceItem
Comparative	TargetItem
Comparative	@ alias
Physical Data Product	PhysicalRecordSegmentUsed
Reusable	@id
Reusable	@agency
Reusable	IdentifyingAgency
Reusable	ID
Reusable	Datum
Structured String Type	
Archive	AvailabilityStatus
Archive	CollectionCompleteness
Archive	ConfidentialityStatement
Archive	Restrictions
Archive	CitationRequirement
Archive	DepositRequirement
Archive	AccessConditions
Archive	Disclaimer
Comparative	ComparisonDescription
Comparative	Commonality
Comparative	Difference
Conceptual Component	HumanReadable
Conceptual Component	Comments
Conceptual Component	Difference
Data Collection	SourceDescription
Data Collection	InstructionText
Data Collection	Characteristic
Data Collection	QuestionIntent
Data Collection	SamplingError
Data Collection	OtherAppraisalProcess
DDIProfile	Description
DDIProfile	Instructions

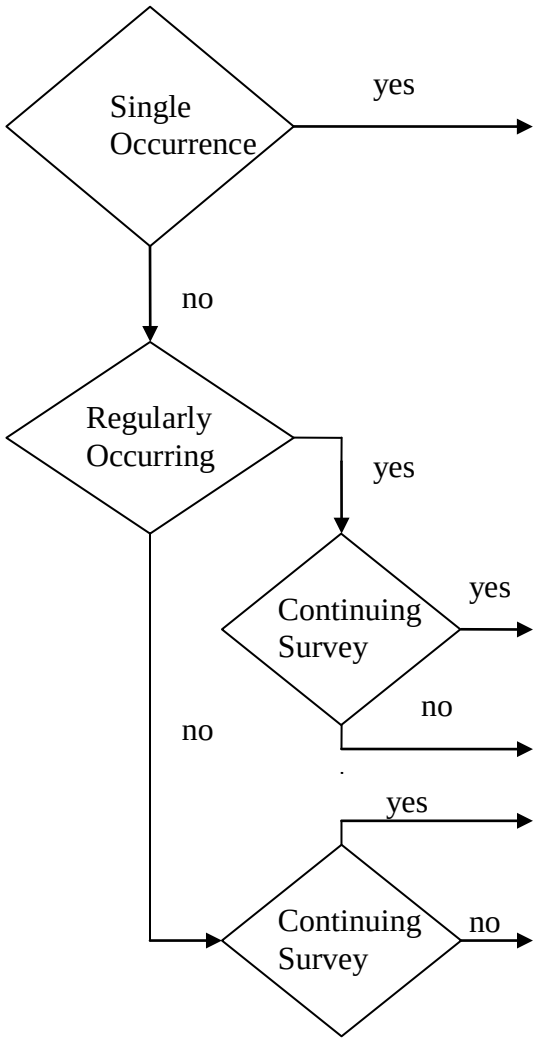
DDIProfile	Description
DDIProfile	Instructions
Logical Product	Definition
Logical Product	VariableDefinition
Physical Instance	ProcessingCheck
Reusable	Content
Reusable	Rationale
Reusable	Description
Reusable	Label
Reusable	Reason
Reusable	User
Reusable	SeriesDescription
Reusable	GeographicLevelDescription

0

Appendix 3: Grouping Attributes and Usage

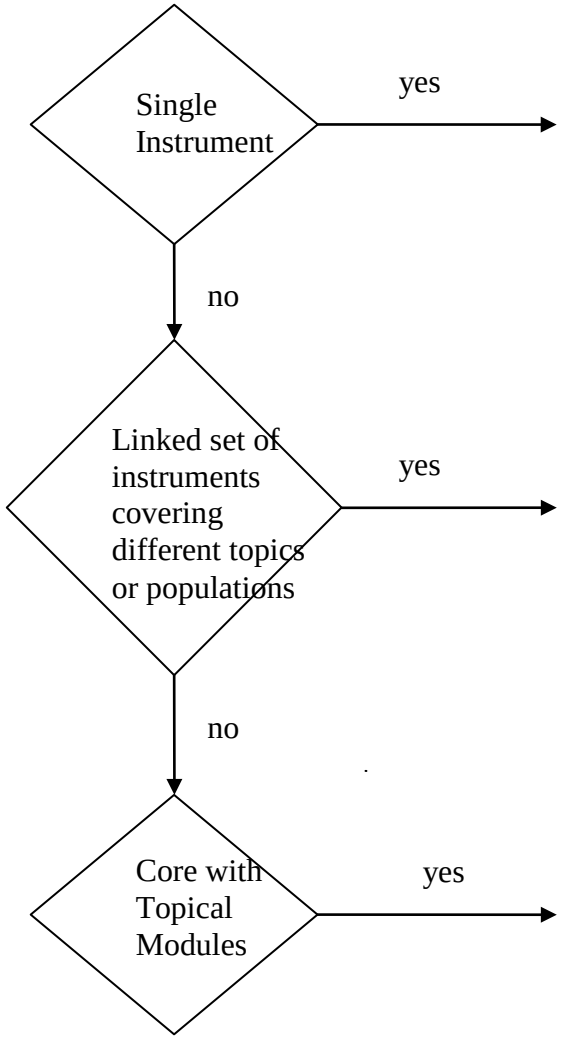
PARAMETER	TAG	DESCRIPTION
TIME	T0	No formal relationship - not a factor of grouping
	T1	Single Occurrence
	T2	Multiple Occurrence: Regular Occurrence: Continuing
	T3	Multiple Occurrence: Regular Occurrence: Limited time
	T4	Multiple Occurrence: Irregular Occurrence: Continuing
	T5	Multiple Occurrence: Irregular Occurrence: Limited time
INSTRUMENT	I0	No formal relationship - not a factor of grouping
	I1	Single
	I2	Multiple: Integrated set of 2 or more instruments used for different subgroups
	I3	Multiple: Base with Topical changes
PANEL	P0	No formal relationship - not a factor of grouping
	P1	Single panel surveyed multiple times
	P2	Single panel surveyed once
	P3	Rolling panel (multiple interviews limited duration)
	P4	Different panel each survey
GEOGRAPHY	G0	No formal relationship - not a factor of grouping
	G1	Single geography surveyed multiple times
	G2	Single geography surveyed once
	G3	Rolling geography (multiple interviews limited duration)
	G4	Different geography each survey
DATA SETS	D0	No formal relationship
	D1	Single data file from a data collection
	D2	Multiple data products from a single data collection
	D3	Integration of multiple data sets into a single integrated structure
	D4	Multiple data files each from a different data collection
LANGUAGE	L0	No formal relationship - not a factor of grouping
	L1	Single language
	L2	All original languages with full language equivalence
	L3	Original language(s) plus translation(s) with full language equivalence
	L4	Translations from external original; full language equivalence
	L5	Translations from a non-included original and have full language equivalence
	L6	Original languages(s) plus translation(s) with partial relationship
	L7	Translations from external original; partial relationships

Note that values ending in '0' denote that the group contains no formal relationships along the given parameter between its children. These values are not shown in the following diagrams.



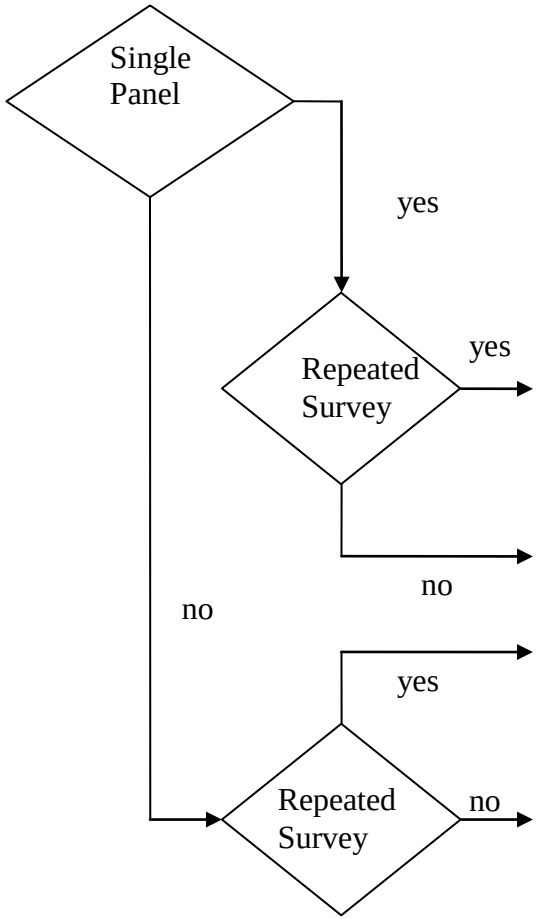
VALUE	EXPECTED INFORMATION	
	In GROUP	Lower in Model [in Data Collection]
T1	Single Data Date Range	
T2	Periodicity Start Date	Single Date Date Range
T3	Periodicity Date Range	Single Date Date Range
T4	Description of occurrence model Start Date	Single Date Date Range
T5	Description of occurrence model Date Range	Single Date Date Range

INSTRUMENT



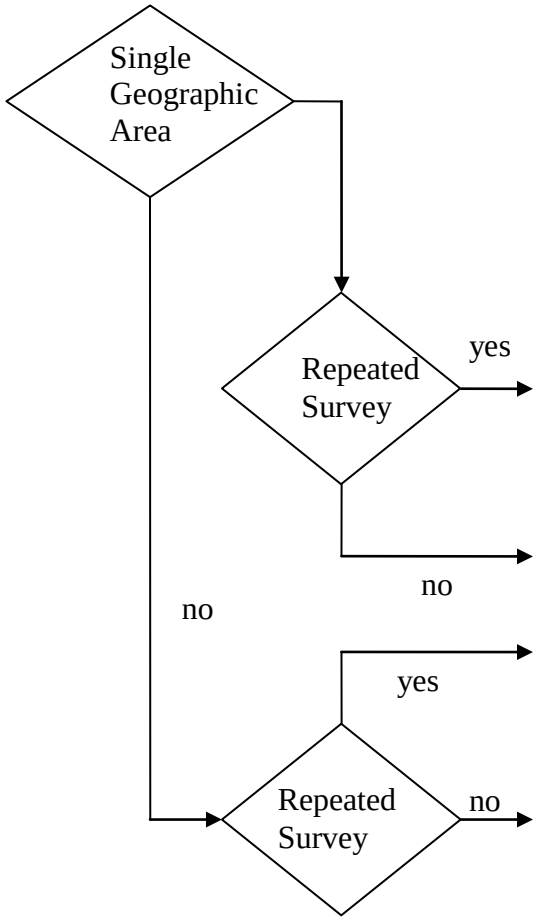
VALUE	EXPECTED INFORMATION	
	In GROUP	Lower in Model [in Data Collection]
I1		Instrument specific information
I2	Number of instruments Relationship of Instruments Linking mechanism	Relationship between instrument and panel(s) Instrument specific information
I3	Variance in periodicity of core and topic modules	Relationship between instrument and panel(s) Instrument specific information

PANEL



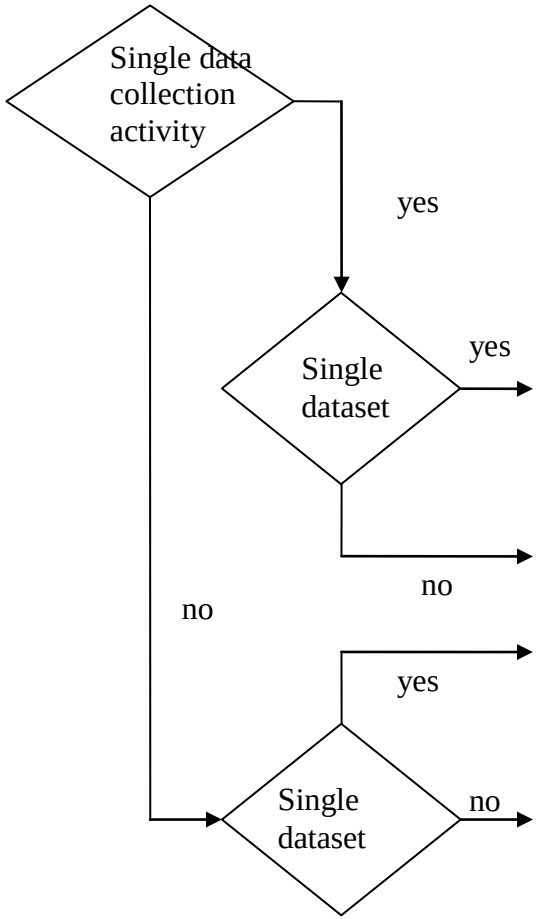
VALUE	EXPECTED INFORMATION	
	In GROUP	Lower in Model [in Data Collection and not covered in other attribute information]
G1	Number of repeats for panel Universe	
G2		Universe Sampling method and size
G3	Overall Universe Wave pattern Number of repeats per panel	Specific Universes Specifics in sampling method and size
G4	Overall Universe Sample selection/ differentiation	Specific Universe Specifics in sampling method and size

GEOGRAPHY



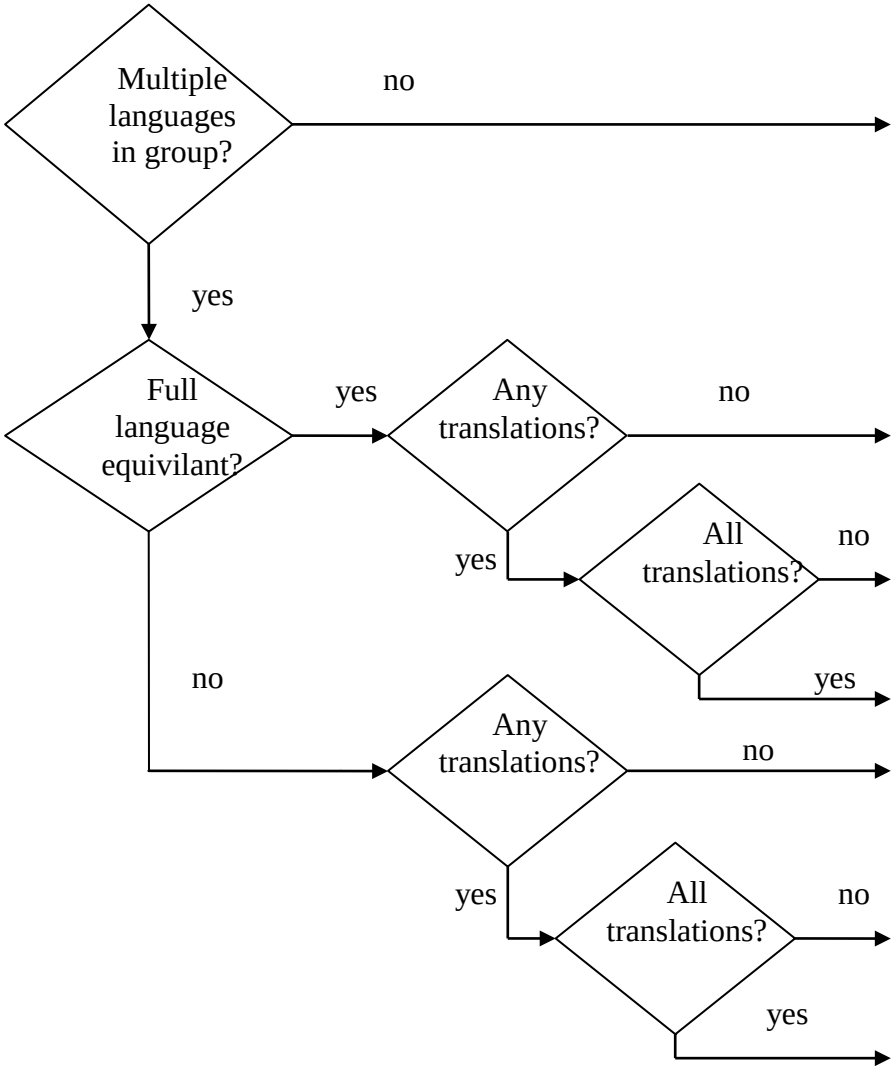
VALUE	EXPECTED INFORMATION	
	In GROUP	Lower in Model [in Data Collection and not covered in other attribute information]
G1	Number of repeats for geography Geographic Cover	
G2		Geographic Cover
G3	Overall Geographic Cover Wave pattern Number of repeats per geography	Specific Geographic Cover
G4	Overall Geographic Cover Geographic selection/ differentiation	Specific Geographic Cover Changes in geographic selection

DATASETS



VALUE	EXPECTED INFORMATION	
	In GROUP	Lower in Model [in Data Collection or lower and not covered in other attribute information]
D1		
D2	Rational for multiple logical products Product mix details	Specific logical data set information [in logical data descriptions with links back to questionnaire]; possible concept keys [logical module}
D3	Integration basis Purpose of integration Selection process Linking process Harmonization process	Specifics of individual data collection processes
D4	Dataset to data collection activity mapping	

LANGUAGE



VALUE	EXPECTED INFORMATION	
	In GROUP	LOWER IN MODEL
L1	single language	
L2	all original languages with full language equivalence	Language identification of study units and lower modules
L3	original(s) plus translation(s) with full language equivalence	Language and translation info at study unit
L4	translations from external original; full language equivalence	Language, translation info and external reference to original
L5	translations from a non-included original and have full language equivalence	Element level language and translation information
L6	original(s) plus translation(s) with partial relationship	Element level language and translation information
L7	translations from external original; partial relationships	Element level lang., translation and reference to original

Appendix 4: XHTML Valid Content

Element Tag	Definition
address	contact information for the document owner or author
blockquote	block quotation, a long quotation set off in a block of text
pre	preformatted text
h1	heading level 1
h2	heading level 2
h3	heading level 3
h4	heading level 4
h5	heading level 5
h6	heading level 6
hr	horizontal line
div	division
p	paragraph
a	anchor which defines the hypertext link using an id attribute
abbr	abbreviation
acronym	acronym
cite	citation
code	computer code text
dfn	definition term
em	emphasized text
kbd	keyboard text
q	quotation, short in line
samp	sample computer code
strong	strong text
var	variable part of text
b	bold
big	big text

i	italics
small	small text
sub	subscripted text
sup	superscripted text
tt	teletype text
br	line break
span	section in a document
dl	list
dt	list term
dd	list definition
ol	ordered list
ul	unordered list
li	list item
table	table
caption	caption
thead	header content in a table
tfoot	footer content in a table
tbody	body content in a table
colgroup	group of columns in a table for formatting
col	attribute values for one or more columns in a table
tr	row in a table
th	header cell in a table
td	cell in a table

Note that these elements exist in the xhtml namespace and must be prefixed with that namespace, e.g. <xhtml:p>
Additional information about XHTML tags can be found at http://www.w3schools.com/tags/default.asp

Appendix 5: Genericode Example

```
<?xml version="1.0" encoding="UTF-8"?>
<gc:CodeList xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xmlns:gc="http://docs.oasis-open.org/codelist/ns/genericode/1.0/"
xmlns:xhtml="http://www.w3.org/1999/xhtml" xsi:schemaLocation="http://docs.oasis-
open.org/codelist/ns/genericode/1.0/ http://docs.oasis-open.org/codelist/cs-genericode-
1.0/xsd/genericode.xsd">
  <Annotation>
    <Description>
      <xhtml:p class="ModuleName">datacollection</xhtml:p>
      <xhtml:p class="Title">Time Method</xhtml:p>
      <xhtml:p
class="XPath">/n1:DDIInstance/s:StudyUnit/d:DataCollection/d:Methodology/d:TimeMethod</xh
tml:p>
      <xhtml:p class="Description">Controlled vocabulary for time method</xhtml:p>
      <xhtml:p class="Description">A new paragraph.</xhtml:p>
    </Description>
  </Annotation>
  <Identification>
    <ShortName>TimeMethod</ShortName>
    <Version>1.0.0</Version>
    <CanonicalUri/>
    <CanonicalVersionUri/>

<LocationUri>http://www.ddialliance.org/ControlledVocabularies/TimeMethod_gc.xml</Locatio
nUri>
    <Agency>
      <LongName>DDI Alliance</LongName>
    </Agency>
  </Identification>
```

```
  <ColumnSet>
    <Column Id="Code" Use="required">
      <Annotation>
     <Description>
       <xhtml:p>Non-hierarchical structure: just the code (no dots are
allowed).</xhtml:p>
       <xhtml:p>Hierarchical structure: the full code, the levels are separated by dots.
Example: Longitudinal.Panel.Continuous</xhtml:p>
       <xhtml:p>Rules for naming a code:<xhtml:ul>
         <xhtml:li>only letters are allowed</xhtml:li>
         <xhtml:li>first letter must be uppercase</xhtml:li>
         <xhtml:li>CamelCase must be used when using multiple words. Example:
TimeMethod</xhtml:li>
         <xhtml:li>only full words are allowed</xhtml:li>
         <xhtml:li>abbreviations can only be used when the acronym is better known than
the full wording</xhtml:li>
       </xhtml:ul>
       </xhtml:p>
     </Description>
      </Annotation>
      <ShortName/>
      <Data Type="string"/>
    </Column>
    <Column Id="ParentCode" Use="optional">
      <Annotation>
        <Description>
          <xhtml:p>Non-hierarchical structure: not used.</xhtml:p>
          <xhtml:p>Hierarchical structure: The full code of the parent. Example:
Longitudinal.Panel</xhtml:p>
        </Description>
      </Annotation>
```

```
      <ShortName/>
      <Data Type="string"/>
    </Column>
    <Column Id="LevelSpecificCode" Use="optional">
      <Annotation>
        <Description>
          <xhtml:p>Non-hierarchical structure: not used.</xhtml:p>
          <xhtml:p>Hierarchical structure: The code on a specific level, no dots are
allowed. Example: Continuous</xhtml:p>
        </Description>
      </Annotation>
      <ShortName/>
      <Data Type="string"/>
    </Column>
    <Column Id="Definition" Use="required">
      <Annotation>
        <Description>
          <xhtml:p>Definition of the code.</xhtml:p>
        </Description>
      </Annotation>
      <ShortName/>
      <Data Type="string"/>
    </Column>
    <Column Id="Caption" Use="required">
      <Annotation>
        <Description>
          <xhtml:p>Caption of the code.</xhtml:p>
        </Description>
      </Annotation>
      <ShortName/>
      <Data Type="string"/>
```

```
  </Column>
</ColumnSet>
<SimpleCodeList>
  <Row>
    <Value ColumnRef="Code">
      <SimpleValue>Longitudinal</SimpleValue>
    </Value>
    <Value ColumnRef="Definition">
      <SimpleValue>Longitudinal</SimpleValue>
    </Value>
    <Value ColumnRef="Caption">
      <SimpleValue/>
    </Value>
  </Row>
  <Row>
    <Value ColumnRef="Code">
      <SimpleValue>Longitudinal.EventBasedCohort</SimpleValue>
    </Value>
    <Value ColumnRef="ParentCode">
      <SimpleValue>Longitudinal</SimpleValue>
    </Value>
    <Value ColumnRef="LevelSpecificCode">
      <SimpleValue>EventBasedCohort</SimpleValue>
    </Value>
    <Value ColumnRef="Definition">
      <SimpleValue>Cohort/Event-based</SimpleValue>
    </Value>
    <Value ColumnRef="Caption">
      <SimpleValue/>
    </Value>
  </Row>
```

```
<Row>
  <Value ColumnRef="Code">
    <SimpleValue>Longitudinal.RepeatedCrossSection</SimpleValue>
  </Value>
  <Value ColumnRef="ParentCode">
    <SimpleValue>Longitudinal</SimpleValue>
  </Value>
  <Value ColumnRef="LevelSpecificCode">
    <SimpleValue>RepeatedCrossSection</SimpleValue>
  </Value>
  <Value ColumnRef="Definition">
    <SimpleValue>Trend /Repeated cross-section</SimpleValue>
  </Value>
  <Value ColumnRef="Caption">
    <SimpleValue/>
  </Value>
</Row>
<Row>
  <Value ColumnRef="Code">
    <SimpleValue>Longitudinal.Panel</SimpleValue>
  </Value>
  <Value ColumnRef="ParentCode">
    <SimpleValue>Longitudinal</SimpleValue>
  </Value>
  <Value ColumnRef="LevelSpecificCode">
    <SimpleValue>Panel</SimpleValue>
  </Value>
  <Value ColumnRef="Definition">
    <SimpleValue>Panel</SimpleValue>
  </Value>
  <Value ColumnRef="Caption">
```

```
      <SimpleValue/>
    </Value>
  </Row>
  <Row>
    <Value ColumnRef="Code">
      <SimpleValue>Longitudinal.Panel.Continuous</SimpleValue>
    </Value>
    <Value ColumnRef="ParentCode">
      <SimpleValue>Longitudinal.Panel</SimpleValue>
    </Value>
    <Value ColumnRef="LevelSpecificCode">
      <SimpleValue>Continuous</SimpleValue>
    </Value>
    <Value ColumnRef="Definition">
      <SimpleValue>Continuous</SimpleValue>
    </Value>
    <Value ColumnRef="Caption">
      <SimpleValue/>
    </Value>
  </Row>
  <Row>
    <Value ColumnRef="Code">
      <SimpleValue>Longitudinal.Panel.Interval</SimpleValue>
    </Value>
    <Value ColumnRef="ParentCode">
      <SimpleValue>Longitudinal.Panel</SimpleValue>
    </Value>
    <Value ColumnRef="LevelSpecificCode">
      <SimpleValue>Interval</SimpleValue>
    </Value>
    <Value ColumnRef="Definition">
```

```
      <SimpleValue>Interval</SimpleValue>
    </Value>
    <Value ColumnRef="Caption">
      <SimpleValue/>
    </Value>
  </Row>
  <Row>
    <Value ColumnRef="Code">
      <SimpleValue>TimeSeries</SimpleValue>
    </Value>
    <Value ColumnRef="Definition">
      <SimpleValue>Time Series</SimpleValue>
    </Value>
    <Value ColumnRef="Caption">
      <SimpleValue/>
    </Value>
  </Row>
  <Row>
    <Value ColumnRef="Code">
      <SimpleValue>TimeSeries.Continuous</SimpleValue>
    </Value>
    <Value ColumnRef="ParentCode">
      <SimpleValue>TimeSeries</SimpleValue>
    </Value>
    <Value ColumnRef="LevelSpecificCode">
      <SimpleValue>Continuous</SimpleValue>
    </Value>
    <Value ColumnRef="Definition">
      <SimpleValue>Continuous</SimpleValue>
    </Value>
    <Value ColumnRef="Caption">
```

```
    <SimpleValue/>
  </Value>
</Row>
<Row>
  <Value ColumnRef="Code">
    <SimpleValue>TimeSeries.Discrete</SimpleValue>
  </Value>
  <Value ColumnRef="ParentCode">
    <SimpleValue>TimeSeries</SimpleValue>
  </Value>
  <Value ColumnRef="LevelSpecificCode">
    <SimpleValue>Discrete</SimpleValue>
  </Value>
  <Value ColumnRef="Definition">
    <SimpleValue>Discrete</SimpleValue>
  </Value>
  <Value ColumnRef="Caption">
    <SimpleValue/>
  </Value>
</Row>
<Row>
  <Value ColumnRef="Code">
    <SimpleValue>CrossSectional</SimpleValue>
  </Value>
  <Value ColumnRef="Definition">
    <SimpleValue>Cross-sectional</SimpleValue>
  </Value>
  <Value ColumnRef="Caption">
    <SimpleValue/>
  </Value>
</Row>
```

```
    <Row>
      <Value ColumnRef="Code">
        <SimpleValue>CrossSectionalAdHocFollowUp</SimpleValue>
      </Value>
      <Value ColumnRef="Definition">
        <SimpleValue>Cross-sectional ad-hoc follow-up</SimpleValue>
      </Value>
      <Value ColumnRef="Caption">
        <SimpleValue/>
      </Value>
    </Row>
  </SimpleCodeList>
</gc:CodeList>
```