Data Documentation Initiative (DDI) Technical Specification

Part II:

User Guide

Version 3.0

April 2008

User Guide for DDI Version 3.0

Version 2:
Date: April 28, 2008
Wendy Thomas, Arofan Gregory, J Gager

Table of Contents

1.0 Overall Structure

1.1 Inline Content vs. Referenced Content

The flexibility of a set of schemas means that you have a number of choices in how to organize your DDI XML instance. Schema contents may be held separately as maintained objects and then included by reference in a parent document or included inline as a single document. Inclusion inline means that a section is incorporated into the parent object in a hierarchical manner, with the elements of each part listed in the main document. Inclusion by reference allows you to create, say, a DDIInstance that references other DDIInstances which contain its studyunit module(s), datacollection module(s), logicalproduct module(s) etc. It is, in essence, a short file of external references, that when resolved, result in the full document. Keep in mind that any decision you make now does not preclude later disassembly of an inline document or assembly of a document containing external references into a single inline document.

Either approach or even combinations of the two are all permissible. The approach taken depends on the anticipated use of the document and the needs of the organization creating it. Issues that one should consider include:

Will any of the parts be used in multiple DDI XML instances?

> For example, the data collection module of a census might be used in over 50 different logical products from public use micro-samples to aggregate data tables. You may or may not want all of these logical products expressed as a single document. If not, having the common parts maintained as discrete objects allows you to use them in multiple documents and still retain the relationships among the documents by their mutual use of a common object.

Is it a simple study that is not anticipated to experience major changes, revisions, or extensions?

> Many of the studies found in archives are simple studies that are the result of academic research. Such studies do not undergo extensive changes as they are created, deposited, distributed, and analyzed. In such cases, a single inline document may be the easiest for the researcher to create.

In the final analysis, it is what is most expedient for the creator, publisher, or archive.

1.2 Top-Level Declarations

All XML documents, whether using DTDs or Schemas, must declare their structures at the head of the file. Because schema structures tend to use multiple sources (multiple schemas), their declaration is both more complex and provides for more options. In essence, the header provides the following information:

Declaration that this is an xml file:
    <?xml version="1.0"?>

The primary schema's primary element tag:
    Left angle bracket
    abbreviation for the schema if used
    colon
    primary element tag
    EXAMPLE:
      <s:StudyUnit

The location of the primary schema including its URN filename and path (note that the path can be to an internal copy of the schemas or an http path to a remote copy):
    xsi:schemaLocation
    equal sign
    open quote
    schema URN
    space
    name of schema file including internal path or http path
    close quote
    EXAMPLE:
      xsi:schemaLocation="ddi:studyunit:3_0 C:\\ddi\schemas\studyunit.xsd"

A namespace is the full URI of a schema or element. In the declaration, an abbreviation or prefix is assigned to the XML schema namespace so that elements from that schema can be uniquely identified with a [prefix]:[element name] (see http://en.wikipedia.org/wiki/XML_Namespaces for additional information and tutorial). All of the schemas required by the document must be identified by namespace and assigned an abbreviation to be used by any element found outside of the parent schema. Abbreviations are assigned to a namespace by the “xmlns:” statement, which declares first the abbreviation and then the namespace of the schema as ddi:<schema name>:version number. For example, xmlns:r="ddi:reusable:3_0"
    xmlns
    colon
    schema abbreviation
    equal sign
    open quote
    schema URN or remote site
    close quote
    EXAMPLES:
      xmlns:r="ddi:reusable:3_0"

```
        xmlns:xhtml="http://www.w3.org/1999/xhtml"
```

All schemas except the top level instance.xsd have been assigned abbreviations as a means of referencing them from within the schema definitions. Using these abbreviations is a convenient convention if you want your documents to be more easily recognizable by other human readers. You may alternately assign your own and some XML editors assign default abbreviations. Systems will follow the abbreviation pattern identified in the header. The following table provides the abbreviation and URN used in the xmlns declarations, and the schema file name used in the declaration of the primary schema.

Abbr	URN ddi:schema:version	Schema name (ddi schemas)
	ddi:instance:3_0	instance.xsd
s	ddi:studyunit:3_0	studyunit.xsd
d	ddi:datacollection:3_0	datacollection.xsd
l	ddi:logicalproduct:3_0	logicalproduct.xsd
c	ddi:conceptualcomponent:3_0	conceptualcomponent.xsd
cm	ddi:comparative:3_0	comparative.xsd
g	ddi:group:3_0	group.xsd
pr	ddi:ddiprofile:3_0	ddiprofile.xsd
a	ddi:archive:3_0	archive.xsd
o	ddi:organizations:3_0	organizations.xsd
p	ddi:physicaldataproduct:3_0	physicaldataproduct.xsd
pi	ddi:physicalinstance:3_0	physicalinstance.xsd
ds	ddi:dataset:3_0	dataset.xsd
r	ddi:reusable:3_0	reusable.xsd
dc	ddi:dcelements:3_0	dcelements.xsd
xhtml	http://www.w3.org/1999/xhtml	[reference to standard]
xs	http://www.w3.org/XML/1998/namespace	xml.xsd
m1	ddi:physicaldataproduct_ncube_normal:3_0	physicaldataproduct_ncube_normal.xsd
m2	ddi:physicaldataproduct_ncube_tabular:3_0	physicaldataproduct_ncube_tabular.xsd
m3	ddi:physicaldataproduct_ncube_inline:3_0	physicaldataproduct_ncube_inline.xsd
m4	ddi:physicaldataproduct_proprietary:3_0_Beta	physicaldataproduct_proprietary.xsd

The first example below shows a document using the instance.xsd as the primary schema. The primary element of instance.xsd is DDIInstance (parent of all other elements in instance.xsd). In this example, the user has declared all the schemas when in fact many are never used in the document. Note that the user has provided the abbreviation “ns1” to the instance schema. This is common when using XML editing software as they tend to assign prefixes for everything. With the exception of this abbreviation, others were taken from the schema declarations themselves. Both examples lack path information for the primary schema, and so the xml document needs to reside in the same directory as the schema. Once DDI 3.0 is officially published, we would normally use the official maintained version of the schema or an agreed upon locally held copy, accessible to the system processing the xml document.

```
<?xml version="1.0"?>
<ns1:DDIInstance
```

```
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xsi:schemaLocation="ddi:instance:3_0 instance.xsd"
xmlns:ns1="ddi:instance:3_0"
xmlns:r="ddi:reusable:3_0"
xmlns:xhtml="http://www.w3.org/1999/xhtml"
xmlns:dc="ddi:dcelements:3_0"
xmlns:a="ddi:archive:3_0"
xmlns:g="ddi:group:3_0"
xmlns:cm="ddi:comparative:3_0"
xmlns:c="ddi:conceptualcomponent:3_0"
xmlns:d="ddi:datacollection:3_0"
xmlns:l="ddi:logicalproduct:3_0"
xmlns:p="ddi:physicaldataproduct:3_0"
xmlns:ds="ddi:dataset:3_0"
xmlns:pi="ddi:physicalinstance:3_0"
xmlns:m1="ddi:physicaldataproduct_ncube_normal:3_0"
xmlns:m2="ddi:physicaldataproduct_ncube_tabular:3_0"
xmlns:m3="ddi:physicaldataproduct_ncube_inline:3_0"
xmlns:m4="ddi:physicaldataproduct_proprietary:3_0_Beta"
xmlns:s="ddi:studyunit:3_0"
xmlns:pr="ddi:ddiprofile:3_0">
```

In the second example the user has declared the studyunit.xsd to be the primary schema and used its abbreviation “s”. The primary element in studyunit.xsd is StudyUnit.

```
<?xml version="1.0"?>
<s:StudyUnit xmlns:s="ddi:studyunit:3_0"
xmlns:c="ddi:concept:3_0"
xmlns:l="ddi:logicalproduct:3_0"
xmlns:r="ddi:reusable:3_0"
xmlns:d="ddi:datacollection:3_0"
xmlns:xhtml="http://www.w3.org/1999/xhtml"
xmlns:xs="http://www.w3.org/2001/XMLSchema"
xsi:schemaLocation="ddi:studyunit:3_0 studyunit.xsd">
```

Note that the use of anything but the DDIInstance module (instance.xsd) as a top-level element is reserved for non-public use of the DDI standard (e.g., within a single organization during processing). Any publication of DDI for general use should always use DDIInstance as the top-level element. This convention insures that all external systems of a single standardized top level structure for all DDI documents.

1.3 Standard Model Contents

Complex Reusable Element	Description of Purpose and Usage
Identification is provided through use of MaintainableType as an extension base	This is the identification of the maintainable module. It must include an attribute id, as well as agency, version and versionDate. All elements within the module can inherit the Agency and Version from this complex element so it must be declared at the highest level.
Coverage	Coverage provides detail for Topical, Temporal, and Spatial Coverage. This should always be stated in the studyunit, and all sub-modules inherit this coverage unless they use this same element to constrain one or more of the coverage areas. A group should reflect the coverage of all studyunits included in the group.
OtherMaterial	OtherMaterial allows you to provide a citation, identifier, and type for any form of material (digital or otherwise) related to the study. This can be linked to any identifiable element in the document. This is available in all major modules so that OtherMaterial specific to a module can be placed at the most appropriate level. All OtherMaterial content is placed together near the top of a module. Storing OtherMaterial in the most relevant module facilitates keeping this information with the related identified objects if the instance is later broken down into its component parts.
Note	Note is a repeatable complex element located near the top of each module. Think of it in the same way you would a set of footnotes at the end of an article. The major features of Note are the ability to type the note and to attach it to any identifiable element in the module. A note can be created once and attached to as many elements as needed. Note was designed to be used during the production process to note questions, cleanups and verifications. Its single location and multiple linkages make it easy to edit or remove.

1.3.1 Coverage

1.3.1.1 Topical Coverage

Topical coverage provides for both subject and keywords. It is an identifiable object. Subject is intended to be a structured subject from a controlled vocabulary such as MESH, LC, etc. Keywords are simple terms, but DDI allows for the creation of a controlled vocabulary to further identify and relate keywords between instances or with large collections.

1.3.1.2 Temporal Coverage

Temporal coverage provides information on the dates covered by the data within the study unit or group. Each date is of the date structure described in section 2.7. At the top level (Study Unit or Group) Temporal coverage should provide inclusive coverage. When used at a lower level, for example a single data file, Temporal coverage can be used to constrain the coverage to a specific time or time period within the defined top level coverage.

1.3.1.3 Spatial (Geographic) Coverage

Geographic coverage provides the following sections:

- A human-readable description of the geographic coverage which maps to the Dublin Core geographic coverage element
- A bounding box (north, south latitudes and east, west longitudes) to support geographic coordinate point search systems
- The lowest form of geographic description (point, line, polygon, or linear ring)
- Spatial object
- Top Level and Lowest Level
- Summary Data Reference
- The relational structure of geographies
- Specific geographic locations

The human readable description should provide as much information on geographic coverage as you would wish to see in a standard bibliographic record. A bounding box consists of a North Latitude, South Latitude, West Longitude, and East Longitude. Longitude has a value in the range of -180 degrees to 180 degrees expressed as a decimal. Latitude has a value of -90 degrees to 90 degrees expressed as a decimal. Bounding boxes provide a first pass identification for coordinate value searches. While the area may seem quite larger than the geography being covered (for example a bounding box of the current United States includes most of Canada), in terms of the entire surface of the earth, it is relatively specific. Internal bounding polygons can provide further coordinate point detail. Geographic search system will also be able to make use of Geographic Structure and Geographic Location details if they make use of known gazetteer names and structure definitions.

The spatial object indicates whether the data is reported for points (a specific address), a line (street, boarder), ring (area within X distance of a point in all directions), or polygon (State, tract, place, block). This is not the level that the data is collected at, but the level it is available in the dataset. This information tells the geographer what he is able to do with the data in terms of aggregation. Points can be aggregated to lines and polygons, but polygon level data cannot be disaggregated to point data. Top level and Lowest level are the equivalents of extent of coverage (for example Europe) and the smallest identifiable area (for example Country). Both may be expressed as Names and as references to specific levels in a GeographicStructure definition.

The last two are new features introduced in DDI 3.0 that allow for specific identification of geographic structures (types of geographic areas like Countries, States, Cities, etc.) and specific geographic locations (for example Germany, France, Ghana, Japan, Canada, etc.). In addition a geographic polygon or reference to shape files can be made at this level. These schemes can be large and complex and the ability to create these once and reuse them via reference was intended to both increase access to this level of geographic information, and to provide a level of comparability by having multiple studies pointing to the same geographic schemes. If this information is provided inline within the DDI Instance, the information is held with GeographicStructureScheme and GeographicLocationScheme found in conceptualcomponent.xsd. The appropriate contents are then referenced by SpatialCoverage.

The purpose of the various geographic coverage descriptions is two-fold. First, it provides information in a format that is more readily understood by geographers and GIS systems. Second it provides access to geographic structure information that was frequently only available by querying the data or by referring to outside documents. Charts and tables providing coding information and relationships generally required human interpretation in order to use them for accessing data. DDI 3.0 attempts to provide this information in a form that can be processed for both data discovery and data exploration. The major components and their functions are listed below. It is important to have a sense of how this information is used by others in order to understand the implications of excluding it from your document.

Element	**Usage**
Bounding Box	The bounding box is the north/south latitudes and east/west longitudes that bound the fullest extent of the geography being covered by the study. While this frequently encompasses more than the intended area (for instance the bounding box for the United States ends up including most of Canada), it is used as a first pass locator, by systems that search the world in terms of a coordinate point. Even a large, overextended box is only a small percentage of the earth's surface and therefore effectively limits the search area for this type of system.

Description	This is the human-readable description of the geographic coverage that maps to the Dublin Core "spatial coverage." It should be as informative as you wish, noting at minimum the extent of the coverage (top level or what the bounding box is bounding) and the smallest level of identifiable geography available (lowest level). Note that while DDI allows structure in this field through the use of xhtml tags, these tags will be lost if the information is transferred to Dublin Core. Make sure content is understandable without the structure elements if you anticipate using this field to populate a Dublin Core record.
Geography Structure Variable	Many data files that use geography as case identification will have a variable that will define the type of geographic level for a specific record (such as Country, county, city, etc.). This is a reference to that variable, which should provide a listing of each geographic type available and its type identification. Even without a full geographic structure description, this single variable can provide a great deal of information about the available geographies within the data file.
Spatial Object	Data are gathered and reported at a variety of object levels. This is one of the basic pieces of information geographers need to know in order to map data. Point data can be assigned or aggregated to lines, linear rings, and polygons, but polygons cannot be separated into their contained points.
Geographic Structure	Geographic structure provides detailed information on the types of geography available and how the various types relate to each other. Single hierarchies, layered hierarchies, and restricted coverage information are provided here.
Geographic Locations	When a file contains a small set of locations or in particular when it contains selected locations of a single type, it is often useful to have that information in the metadata. For example, does a file containing data on cities of 25,000 or more include the city of Black Duck, Minnesota? Geographic

	locations may contain a more detailed bounding polygon description and/or pointers to external geographic shape or boundary files.
Summary Data Reference	If the geographic structure is defined, this element can be used to identify all levels that have summary data attached to them. For example, a file with the geographic structure of State/County/Tract may have data records for all levels or only for the lowest level (higher levels are created through aggregation).
Top Level Reference	This identifies the broadest area of coverage by text name and also optionally as a reference to the geographic level that describes it.
Lowest Level Reference	This identifies the smallest area of coverage by text name and also optionally as a reference to the geographic level that describes it. This is similar to the geographic unit as described in earlier versions of DDI.

The Geographic Structure describes the levels of geography and their relationships by defining a level and its parent or parents. Note that coverage limitation information can be included at specific levels to further define the coverage structure. This information will help external systems identify when only a subset of locations are available in the data (such as only Counties with 100 or more farms of $1,000 or more in yearly income) and the type and location of geographic codes in the data.

```
<r:Geography isIdentifiable=”true” id=”GEO_0”>
        <r:Level>
                <r:Code>010</r:Code>
                <r:AurhorityOrganziationReference isReference=”true” isExternal=”false”>
                        <r:ID>USCB</r:ID>
                </r: AurhorityOrganziationReference>
                <r:Name>Country</r:Name>
        </r:Level>
<r:Geography>
<r:Geography isIdentifiable=”false” id=”GEO_1”>
        <r:Level>
                <r:Code>040</r:Code>
                <r:AurhorityOrganziationReference isReference=”true” isExternal=”false”>
                        <r:ID>USCB</r:ID>
                </r: AurhorityOrganziationReference>
                <r:Name>State</r:Name>
        </r:Level>
```

```
      <r:ParentGeography isReference=”true” isExhaustiveCoverage=”true”>
            <r:ID>GEO_0</r:ID>
      </r:ParentGeography>
<r:Geography>
<r:Geography isIdentifiable=”false” id=”GEO_2”>
      <r:Level>
            <r:Code>050</r:Code>
            <r:AurhorityOrganziationReference isReference=”true”  isExternal=”false”>
                  <r:ID>USCB</r:ID>
            </r:AurhorityOrganziationReference>
            <r:Name>County</r:Name>
             <r:CoverageLimitation>Counties with at least 100 farms with $1,000 or in
            more yearlyincome</r:CoverageLimitation>
      </r:Level>
      <r:ParentGeography isReference=”true isExhaustiveCoverage=”true”>
            <r:ID>GEO_1</r:ID>
      </r:ParentGeography>
<r:Geography>
```

The complementary piece to this is Geographic Locations where explicit locations can be listed. This is useful for files with limited geography or where shape files will be attached to specific geographic locations. The description starts with information on a specific type of geography which indicates if there is any coverage limitation, and then identifies the variable containing the geographic code for that area type and the authority reference for the coding system. This is followed by a list of specific location codes. Note that specific codes do not need to be included, so that you can provide all of the information noted above without going into further detail.

Individual location listings provide a specific geographic code, a name, and the geographic time (valid time period for the geography being used as this does not always match that of the data), as well as the option of including a bounding polygon (and excluding polygon) and/or reference to a shape file to describe the area in as detailed a manner as desired.

```
<r:GeographicLocation isVersionable=”true” id=”GL_1”>
  <r:AurhorityOrganziationReference isReference=”true” isExternal=”false”>
        <r:ID>USCB</r:ID>
      </r: AurhorityOrganziationReference>
    <r:GeographicLevelRefereence isReference=”true”>
    <r:ID>GEO_2</r:ID>
  <r:GeographicLevelRefereence>
  <r: Values>
    <r:VariableReference>
      <r:Reference><r:ID>STCNTY</r:ID></ r:Reference>
    </r:VariableReference>
    <r:GeographyValue>
      <r:GeographyCode>
         <r:Value>2700010</r:Value>
      </r:GeographyCode>
      <r:GeographyName>Aitkin [MN]</r:GeographyName>
        <r:GeographicTime>
      <r:StartDate>1860</r:StartDate>
      <r:EndDate>9999</r:EndDate>
        </r:GeographicTime>
        <r:BoundingPolygon>
          <r:ExternalURI>http://data.nhgis.org/MNcty1860.prj</r:ExternalURI>
          <r:PolygonLinkCode>2700010_1860</r:PolygonLinkCode>
          <r:GeographicTime>
            <r:SimpleDate>1860</r:SimpleDate>
          </r:GeographicTime>
        </r:BoundingPolygon>
    </r:GeographyValue>
    <r:GeographyValue>
      <r:GeographyCode>
        <r:Value>2700310</r:Value>
      </r:GeographyCode>
      <r:GeographyName>Cook [MN]</r:GeographyName>
      <r:GeographicTime>
        <r:StartDate>1880</r:StartDate>
        <r:EndDate>9999</r:EndDate>
      </r:GeographicTime>
    </r:GeographyValue>
  </r:Values>
</r:GegraphicLocation>
```

2.0 Technical Structures

DDI 3.0 is dependent upon creating relationships by reference. While a hierarchy may seem more intuitive to a user, a strict hierarchical approach limits the availability of materials for reuse. Reuse facilitates the development of documentation throughout the life cycle of the study, reduces the possibility for human entry error, and provides a basic level of implicit comparability. In order to provide interoperability for reference systems (allowing for exchange and reuse of existing metadata) a consistent form of identification must be employed.

2.1 Identifiable Objects

Not all elements are identifiable, but identification, when applicable, is required. If we want DDI metadata to be shared and act as a transport structure to run programs and processes, consistent use of identifiers is essential. Non-identified elements generally require context and “live” within complex elements that are identifiable and provide context for the non-identified element.

Elements that have identification only, that are not versionable, inherit their version from their versionable parent, and their agency from their maintainable parent agency. In other words, if the version or agency of the parent element changes the identifiable element is considered to be part of this new version.

Elements that are identifiable only use the complex element r:IdentifiableType as an extension base. This provides it with a consistent set of attributes that are used to identify the element and to note any local changes for inherited contents. The element may also be assigned a Name which can be repeated for language and/or geographic alternatives. All identifiable elements must have an ID entered in the attribute id. The full identification can also be expressed as a URL which includes the complete path name. See Part I Appendix 1 for the full URL paths for all identified objects in DDI 3.0. All identifiable elements within a DDI instance can be located by the required fixed attribute isIdentifiable="true" and contain the following element and attributes (attributes are identified with the prefix @ and all start with lower case letters):

Element / attribute	**Description of use**
@id	Required identifier for the element. This MUST be unique within the parent maintainable.
@urn	An optional urn for the element. Note that if there is conflict between the id and urn content, the urn takes precedence.
Name	An optional repeatable element which allows for a human-readable name to be attached to the parent element. It can be repeated for language and/or geographic alternatives.
@action	Action has a controlled vocabulary of "Add", "Update", or "Delete". It is used to identify local overrides to inherited content. • ***Add*** – the element has a unique id and should be used in addition to the inherited elements • ***Update*** – the element has the id of the inherited element which it updates (for example a local Name or label change) • ***Delete*** – the element has the id of the inherited element which is NOT used in the local instance (for example a ProcessingEvent was not used). Note that if the identified element is complex, the entire contents of the complex element will be considered as deleted.

Note that the attribute action is used only with inherited materials. Inheritance occurs with grouping. These action statements provide local overrides for the current inheritance, they themselves cannot be inherited. Note that if the element that contains the change is not identifiable, its parent identifiable should be entered in full including the changed information.

2.2 Versionable Objects

A subset of identifiable elements is also versionable. These are elements for which changes in content are important to note. They use the extension base of VersionableType and include all of the elements and attributes of IdentifiableType. They are identified in a DDI instance by the fixed attribute isVersionable="true". In addition Versionable elements allow one to indicate a major and minor version, the date of the version, who changed it, and why it was changed. If a change occurs at a lower level, it requires a version change of its parent. Users need to understand if the change that was made will affect their analysis of the data. To do this, they need to know why a change was made and who made it. Versionable elements include the following elements and attributes in addition to those listed under Identifiable.

Element / attribute	**Description of use**
@version	If this is missing the default is assumed to be version 1.0. This attribute can contain only numbers and the separator "." But can extend to as many levels of specificity as needed by the maintenance agency. The first number to the left of the separator is the major version number. All subsequent numbers are considered minor versions.
@versionDate	This is the date that the specific version becomes active. It is a simple date containing a dateTime, date, YearMonth, or YearMonth. It should be as specific as possible.
VersionResponsibility	Do not use this to indicate the name of the Maintenance Agency as this is the only agency which can make changes in the content of the instance. VersionResponsibility was provided to allow for the identification of the person or suborganization within the maintenance agency that made the change. This may be important to internal management.
VersionRationale	This provides the reason for the change. The correction of a typographical error may have different ramifications for the end user than the replacement of erroneous content.

Note that versioning is only required for published materials. Maintaining version identification during the production process will depend on the needs of the organization producing the document.

2.3 Maintainable Objects

There are a number of complex elements that represent major blocks of objects that can be maintained outside of a DDI Instance (published as separate entities). All maintainables are published within a DDIInstance under one of the following packaging structures: StudyUnit, Group, or ResoursePackage. Note that a PhysicalInstance can only be published within a StudyUnit or Group. Major modules can be maintained as can all complex elements whose name ends in "Scheme", plus occasional additional complex elements. Note that ResourcePackage and Group are the two possible top level elements of the module scheme Group. Most module schemes contain a single top level containing element which carries the name of the module. Version changes can only be made by the maintaining agency (or at their specific request) so that if a non-maintaining agency makes a change.

Modules	**Scheme**	**Additional**
DDIInstance	QuestionScheme	Instrument
Archive	CategoryScheme	ResourcePackage
StudyUnit	CodingScheme	Group
DataCollection	VariableScheme	
LogicalProduct	ConceptScheme	
PhysicalDataProduct	UniverseScheme	
PhysicalInstance	OrganizationScheme	
Comparative	GeographicStructureScheme	
ConceptualComponent	GeographicLocationScheme	
DDIProfile	NCubeScheme	
	PhysicalStructureScheme	
	RecordLayoutScheme	
	ControlConstructScheme	
	InterviewerInstructionScheme	

Maintainable objects can be used to assemble an instance by reference or to share commonly used sets of information. Elements that are maintainable are extensions of the complex element r:MaintainablType. They include all the elements and attributes of VersionableType and add the attribute agency. The content of Agency is the Name token of maintenance agency. This unique token must be declared in the document as a full organization or individual description or as a reference to an external registry of organizations. Note that the id of all maintainable objects MUST be unique within the maintaining agency.

2.4 Constructing an Identification

The complex elements IdentifiableType, VersionableType, and MantainableID are used to identify described objects for the purposes of internal and/or external referencing. All identification allows for two ways of expressing an entity's identification. The first is through the use of a URN. The URN provides the element type, the maintaining agency, version, and element ID. While URN is recommended, the alternative is using a series of elements that provide the same information as found in the URN. If both URN and a combination of elements are used, the URN takes precedence. The DDI uses a specifically structured URN [see URN part 2.5]. Note that VersionableTypes inherit their IdentifyingAgency from their parent Maintainable object, while the IdentifiableType inherits both its versioning information and IdentifyingAgency from its parent Versionable object. Any object that is published (a Group, StudyUnit, or any Maintainable Object published for inclusion by reference, must be published within a DDIInstance. The ID of this instance MUST be unique within the maintaining agency.

Identification contains a URN and/or the sequence of elements shown below. In addition Identification has an optional repeatable element Name. This is a human-readable name given the entity being identified. This may be a mnemonic that is commonly related to the element. It may be repeated to provide language and/or geographic alternatives.

ELEMENT/ATTRIBUTE	**TYPE**	**Description**
@id	NCName	ID assigned by an agency. This must be a unique identifier within the maintained object. It must start with a character and can contain any character or number plus any of the following non-alphanumeric characters: “*”, “@”, “_”, “$”, or “-“
@agency *(MaintainableType only)*	NCName	The agency maintaining the identification. This is optional if inherited from a parent object. The published instance must contain an entry in an OrganizationScheme either inline or by reference. The content should be the ID of the organization/individual.
@version *(VersionableType and MaintainableType only)*	string	Version number, expressed as a two-part numeric string composed of two positive integers separated by a period. The first number indicates a major version, the second a minor one: 1.0. Optionally, a third integer may indicate sub-version: 1.0.2.
@versionDate *(VersionableType and MaintainableType only)*	BaseDate	Date the version took effect expressed as dateTime, date, YearMonth, or Year
VersionResponsibility *(VersionableType and MaintainableType only)*	string	Reference to the person and/or organization responsible for the version change. This is primarily intended for internal use and can be as detailed as the organization requires.
VersionRationale *(VersionableType and MaintainableType only)*	string	Textual description of the rationale/purpose for a version change. This should be informative to users so that they can determine if the change has potential impact on their analysis.

2.5 URN Structure

A DDI URN has a specific structure that must be followed for uses of the URN in identification and reference to DDI objects.

urn="urn:ddi:3_0:<Maintainable Object Class.Object Class>=<Agency ID>:<ID of maintained object>[<Major Version>.<Minor Version>].<ID of contained object>"

An example of the URN structure for a versionable object is broken down below:

urn:ddi:3_0:VariableScheme.Variable=ICPSR:VScheme_4[1.0]AGE_3[1.0]

- The delimiters are as follows
    - Top level field separator :
    - Hierarchical separator .
    - Object class to identification separator =
    - Version separator [ ]
- All sections are required.
- All published elements are either maintainable or contained within a maintainable object. The maintainable object must be identified first, as the ID is unique within the maintainable. The object type identification will always contain a maintainable object plus the optional specific object if it is either Versionable or Identifiable
- The full path of ids with version where applicable should be provided (Part I Appendix 1 contains an alphabetical listing of identifiable and versionable elements in alphabetical order with their complete nesting string up to parent maintainable)

Breakdown of the example URN:

Identify that this is a URN	urn
Top level field separator	:
It is a DDI URN	ddi
Top level field separator	:
It is DDI version	3_0
Top level field separator	:
The maintainable object type	VariableScheme
Hierarchical separator	.
The object	Variable
Object class to identification separator	=
The maintenance agency is the DDI Alliance	ICPSR
The name of the VariableScheme is	VScheme_4

The version number of this element is encased in version separator	[1.0]
Hierarchical separator	.
With the ID of object	AGE_3
The version number of this element is encased in version separator	[1.0]

Note that the Variable must have the same maintenance agency as the maintainable it resides in, and therefore this is not repeated.

If the element being referenced was a maintainable itself, then the URN would end after the first version element.

Examples

URN of a maintained object

To identify of a variable scheme in DDI 3.0 via a URN would be as follows: **urn="urn:ddi:3_0:VariableScheme=ICPSR:V_GENDER_SCHEME[1.0]".**

URN of an versionable object

All versionable objects are contained within maintainable objects. To identify of a variable in DDI 3.0 via a URN would be as follows: **urn="urn:ddi:3_0:VariableScheme.Variable=ICPSR:V_GENDER_SCHEME[1.0].Male [1.0]"**

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”>
        <Methodology isVersionable=”true” id=”Meth_Type_1” version=”1.0”>
                <TimeMethod isIdentifiable=”true” id=”TM_1”>
```

To identify the identifiable object in the above hierarchy in DDI 3.0 via a URN would be as follows: **urn="urn:ddi:3_0:DataCollection.TimeMethod=ICPSR:DC_5698[2.4].Meth_Type_1[1.0].TM_1"**

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:3_0:OrganizationScheme.Individual=ICPSR:OS_1[1.0].UMICH[1.0].ICPSR[1.0].J_Doe[1.0]”

2.6 Referencing

DDI 3.0 contains four types of references:

Reference	The basic structure used to reference DDI objects
SchemeReference	A special extension to Reference for referencing DDI schemes
OtherMaterial	The basic structure for referencing external non-DDI objects
Image	Reference to an external image file currently used only in the OrganizationScheme

References to DDI objects are all based on the ReferenceType as described in reusable. Elements within a DDI instance which reference DDI objects will all be identified with the fixed attribute isRefrence=”true”. Note that when referencing an object internal to the DDI instance you may simply provide the ID as long as it is unique within the instance. However, if this instance is later parsed into its constituent parts, the parser will need to supply the missing components to uniquely resolve the reference. If the ID of an object is only unique within its maintainable parent (module or scheme), the identification of that maintainable parent must be included to provide a unique identification. Once again, if both a URN and ID are provided, the URN takes precedence if they have conflicting content. Reference includes the following structure.

Element/Attribute	**Description**
Module	This is a complete reference structure for the parent module. This must be used in cases where there have been local modifications. [optional]
Scheme	This is a complete reference structure for the parent scheme. [optional]
URN	A DDI structured URN may be used in addition to or instead of an ID sequence.
ID	ID of the element being referenced. This is the minimal required content for a reference in terms of content and length.
IdentifyingAgency	The NCName of the maintenance agency used as part of the ID sequence [option]

Version	The version number of a referenced versionable or maintainable object used as part of the ID sequence [optional]
@isExternal	The default setting is “false”, indicating that the element will be found inline within the DDI instance. If this value is set to “true” (reference to an external DDI object) the URI must be provided.
@URI	This is the URI for the external location when isExternal is set to “true”
@isReference	Fixed Boolean value of “true”
@lateBound	A Boolean attribute with the default setting of “false” indicating that the reference is to a specific version of the object. When changed to “true” the reference will retrieve the most recent version available. Note that while identifiable objects do not have a version themselves, the object will be obtained from the most recent version of its parent versionable or maintainable object.

The use of late bound and early bound references provides flexibility in the material obtained by the referernce. Early bound (@lateBound=”false” means that you are referencing a specific version and that none other will do. This allows you to replicate the metadata as it was used in a previous analysis. Late bound allows you to request the most recent version of the referenced element. For example, you may always wish to include the most recent version of an Archive module to receive the most current updates of organizational information, life-cycle events, related publications, and access information. Or you may want to be sure you have the most current corrections and updates to the information in a logical product.

SchemeReference extends Reference and provides an addition element set that allows you to exclude one or more identifiable elements from the scheme being included by reference. It allows a user to include say a VariableScheme but exclude one or more variables within that scheme. Most schemes will already allow for selective inclusion of component items from other schemes, but this particular reference simplifies the process when the uses wishes to include the majority of the scheme, excluding only a few items. The additional element Exclude contains an ID and Version element pair where the version is optional. Since the parent maintainable scheme has already been identified, this is all that is needed to identify specific items for exclusion.

OtherMaterial as a reusable is available in all major modules. References to related materials, physical objects, etc. are listed here and can be attached to any identifiable object. This material is not intended to be processed by DDI but is supplied to inform the end-user about non-DDI materials related to the development, collection, analysis, or other materials related to the DDI Instance. OtherMaterial contains the following:

Element/Attribute	**Description**
Citation	Full citation element for the external object
ExternalURLReference	Location of an external electronic object via a URL (for example http://icpsr.umich.edu/)
ExternalURNReference	The namespace of the external object expressed as a URN (for example urn:isbn:0-395-36341-1 the unique International Standard Book Number)
Relationship	This contains both a reference to an identified DDI object and RelationshipDescription (optional/repeatable). Relationship can be repeated to link to multiple DDI Objects.
MIMEType	A standard internet MIME type for use by processing applications
@type	A required type code for type of OtherMaterial.

External objects that are not DDI require the more complex structure of OtherMaterial. The use of non-DDI references is quite limited outside of the standard section for OtherMaterial listings within each of the major modules. Currently use is restricted to ExternalInterviewerInstruction, ExternalAid, and r:OtherMaterial in Generation. ExternalInterviewerInstruction provides the ability to reference non-DDI structured interviewer instructions from with a ControlConstruct for a questionnaire/instrument. It consists of OtherMaterial and an attribute displayText which indicates whether the information should be displayed or simply made accessible (via reference or specific request for display). ExternalAid is available in QuestionItem, MultipleQuestionItem, and ControlConstruct. Its purpose is to provide access to an image, sound, media, or physical object used in the question. For example “Watch the 30 second ad and indicate if you …..” where ExternalAid would be the link to the advertisement. ExternalAid is type=”r:OtherMaterial”. Generation is a structure to describe how a specific Category within a CategoryScheme was generated and includes a child element r:OtherMaterial. An example of how this might be used is a reference to a printed table of Poverty breakpoints for the United States definition of poverty. The Generation would describe how the breakpoints were generated and then provide a link to the resultant table.

2.7 Date

The DDI provides a structure for a Date element which allows a choice between single, simple dates or date ranges. If a date element contains a range, Cycle may be used to indicate occurrence of this range within a series of ranges. An integer is used to identify the cycle. Dates are required to be expressed as ISO 860-formatted dates for all fields. The optional attribute calendar on any parent date item, such as PublicationDate, allows you to designate a non-standard calendar type. Historically-formatted dates may be included in addition for archival or other purposes.

2.7.1 Simple Date

```
<r:PublicationDate calendar=”Georgian”>
<r:SimpleDate>2007</r:SimpleDate>
</r:PublicationDate>
```

This is a single point in time and conforms to any of the following ISO 8601 standard structures.

dateTime	YYYY-MM-DDThh:mm:ss	1982-01-05T23:05:15
date	YYYY-MM-DD	1982-01-05
gYearMonth	YYYY-MM	1982-01
gYear	YYYY	1982
duration	PnYnMnDTnHnMnS	P26Y2M22DT11H5M20S

Note that the “T” in dateTime is literal, denoting the beginning of the Time section, and that “ss” can contain decimals. Optionally, dateTime can be extended by a timezone offset of “Z” to represent Zulu time or GMT. For example, Eastern Standard Time is Z-4.

Note that the “P” in duration is literal and indicates that this is a Period of duration. The other upper case letters are also required with the preceding number providing the number of years (nY), months (nM), etc. A period may be of negative duration, for example a period of minus 10 days (-P10D), by preceding the “P” with a negative sign.

2.7.2 Date Range

A range is expressed as a StartDate and EndDate each expressed in the same format as a simple date. The dates are assumed to be inclusive. The position of this range within a series of ranges is expressed as an integer in Cycle. For non-standard calendars an attribute calendar allows specification of the calendar used.

```
<r:Date>
<r:StartDate>2006-04-01</r:StartDate>
```

```
<r:EndDate>2007-03-31</r:EndDate>
</r:Date>
```

2.7.3 Historical Dates (expressed in formats other than ISO 8601)

All dates can optionally be expressed in other historical structures with an attribute to describe the structure being used. This is simply a string containing the historical date and an attribute historicalDateFormat used to specify the non-ISO date format. For example:

```
<HistoricalDate historicalDateFormat=”Month DD, YYYY”>January 5, 1982</HistoricalDate>
```

Historical date information parallels the simple date, start date and end date structures of the standard DateType.

2.8 String Types

String or text entries have a number of formats to support language differences, the need for structured text, and constraints on content.

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 IdentifableType and 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) . (See section 5.1 for details of use)

The following grid shows which features are available for each type other than NCName. 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

2.9 DDI Profiles

Many DDI users first determine which elements they will use, or not use, and determine any constraints on how they are used for their specific local needs. This is particularly common in organizations who wish to make sure elements are used in a consistent manner, or need information expressed in a specific way to support their systems. Common examples of this are the DDI Core Elements, the CESSDA list of required elements for inclusion in their search system, or NHGIS rules for special use elements. Each of these examples imposes additional required elements, informs the user what items are expected in submitted documents, restricts what is handled by a system, or specifies accepted usage of an element within a system.

DDI profiles allow this type of specification to be defined in a consistent way that can be published and used for validation. This profile describes which DDI elements and attributes are used in a particular use profile or supported by a DDI-conformant application. It uses XPath expressions to identify the used or unused fields in terms of the full possible DDI instance. Its construction is quite simple, but it allows an organization to require certain elements for a DDI instance to be included in a collection (example, CESSDA), to inform contributors of system limitations (example, does not handle NCubes), or to inform an organization on the proper use of elements for internal compliance.

Object	**Purpose**
Identification	Identifies the DDI Profile.
XPathVersion	Provides the version of XPath used. Currently values are either 1.0 or 2.0.
DDINamespace	This is the DDI version, currently 3.0
Used	A repeatable complex element that describes a DDI element used by the profile. This complex element contains an XPath, which points to the element or attribute that is being used. All sub-elements of a used element are assumed to be supported unless explicitly addressed by the profile. The number of supported repetitions may be included in the XPath. An attribute “required” with a default value of “false” allows for requiring elements that are optional in the DDI schema. Note that if a field is required by the DDI Schema, it cannot be made optional.
NotUsed	A repeatable complex element that describes a DDI element NOT used by the profile by providing its XPath. A required DDI Element cannot be disallowed.

Used allows the identification of an AlternateName for the object, a Description and Instructions for use. The attributes include required for changing an optional object to a required object, path, defaultValue to designate a default value if the object is missing, and a fixedValue indicating that the defaultValue noted in the earlier attribute is actually a fixed value.

3.0 Capturing the Background Information

3.1 Study Unit

The first stage in capturing information for a study is always within the study unit. The study unit captures information related to the idea behind the study, the early stages of proposal or project development, and provides it with an identification and basic information on scope and coverage. Remember that this is preliminary content and can be altered as the study progresses through the life cycle. DDI 3.0 was designed to be used for capturing information across the life cycle process of the study (as well as after the fact) and can be used to hold drafts, ideas, and notes that will be formally incorporated at a later date. Remember that once “published,” any changes must be versioned.

Start a DDI xml instance using studyunit.xsd. In essence this section captures content from the inception of the study, from the idea. Include references to all other schemas you may be using, but at minimum include the following xmls identifiers:

.xsd file	Reason for inclusion
studyunit	This is the base source xsd.
reusable	All schemas use elements from reusable.xsd.
conceptualcomponent	Concept, Universe, and Geography definitions are some of the earliest sets of information created.
archive	Captures information on your organization and process and identifies the maintaining agency.
datacollection	Allows you to capture information on planned methodology such as sample design.

Complete the Identification and use all the fields (id, agency, and version) as these can be inherited by elements lower in the structure.

Under Citation provide at least a working Title and Creator.

Use Abstract to capture an outline of the study. Remember that this can be edited later to relate further refinements. The null hypothesis or other description of why the study is being conducted (intended use) should be placed in Purpose.

Complete the upper level or levels of Universe in conceptualcomponents and provide a link from the study unit to the uppermost level of the universe (see Universe, Part 3.5). This is also a good point to begin collecting and structuring your concepts (see Concepts, Part 3.6). These two sections are needed by later modules for reference purposes. While they can be edited and expanded at any point, completing them at this point assists you in clarifying both the population being studied and the concepts covered by the study. The GeographicStructureScheme and GeographicLocationScheme can be started now if known. If external ResourcePackages are available which contain a standard geographic structure and location coding system, it is worthwhile to note it now by referencing those schemes. This information will inform how questions will need to capture geographic information for the study. Include your unit of analysis in AnalysisUnit.

Coverage at this level is the overall coverage of the study. You must define the coverage at this level, as it is inherited or constrained by all lower levels. Lower levels can constrain the coverage by limiting the topical, temporal, or spatial coverage of a specific module, but they cannot expand the coverage. Coverage is made up of the following sections and at least the minimal level information should be provided.

COVERAGE	PRIMARY TAGS	PURPOSE OF INFORMATION
Topical	Subject, Keyword	Provides structured subject headings and unstructured keywords
Temporal	ReferenceDate	Reference dates provide information on dates covered by the data that are outside of the collection period, such as residence 5 years ago.
Spatial	Description, SpatialObject, TopLevelReference, LowestLevelReference	While this may be expanded later in the life-cycle of the study, these are the minimal level recommended tags. Description is the equivalent of the Dublin Core geographic coverage. Spatial Object at this point is the most discrete level at which data will be captured (point, line, polygon, or linear ring). Top Level and Lowest Level are the broadest area of spatial coverage (for example, Europe) of the study, and the most discrete identifiable level (for example, a NUTS 3 or a housing unit). These three pieces of information are used by geographers to determine the usability of the contents for various GIS uses as well as providing basic geographic

		structural information to the data user.

Begin building the OtherMaterial section of the studyunit at this point. Use it to capture citations on materials used as references in developing the study proposal. Provide a citation for any proposals submitted to funding agencies and any other related material. Some of this may be incorporated inline at some point, but you will still want to retain the citation to the separate document.

“Note” can be used to capture information you may want to include in other areas at a later date, but want to hold in a temporary area of your instance. Select an appropriate type attribute so that at a later date these will be easy to locate, attach to appropriate elements, or transfer information to another location and delete.

3.2 Concepts

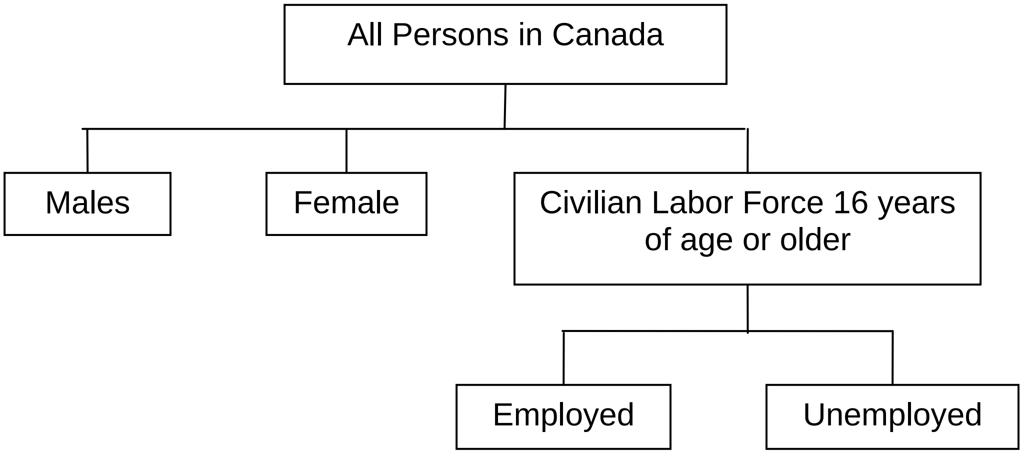
Concepts for variables and questions are described in a concept scheme and referenced by the variable or question. A question can reference multiple concepts; however, a variable must be linked to a single concept. Variables that capture a number of concepts require the creation of a composite concept that captures both concepts under a single concept description. This tends to occur only in some legacy aggregate data files which nest a number of concepts in a non-regular hierarchy. For example “Household Type by Presence and Age of Related Children” would require a concept that includes both a concept for the household type and one or more to capture presence and age of related children. Alternatively, the variable can be split into its component parts (separate concepts) and reassembled within an NCube that defines those sections without data (see NCubes Section 7.5 for details).

In terms of a complete variable description, the concept provides the property of the object in a complete ISO/IEC 11179 description. DDI uses the variable to link the object (universe) with the property (concept) and the representation. A ConceptScheme with just a collection of concepts requires the use of the variable to tie these pieces together. Concept scheme can also contain data element concepts which provide these links internally so that a complete ISO/IEC 11179 description can be held as a resource package outside of the context of a dataset and its variables. A concept is versionable and contains a label, description, and information on similar concepts (reference to the similar concept and a declaration of the difference between the two). A concept may be identified as a “characteristic” through the use of a Boolean attribute. This attribute must be set to “true” if it is referenced by a data element concept.

A data element concept is a concept in the ISO/IEC 11179 sense. It contains a label and description and can identify similar concepts like a general concept. However, it references another concept which has its “isCharacteristic” attribute set to “true” as well as referencing a universe. In this manner it is stating that the data element concept defines a concept that is characteristic of a specific universe.

3.3 Universe

A universe scheme contains a structured listing of all the universes within a study. The top level should reflect the complete universe of the study, for example, “All persons and housing units in Canada”. Lower level subsets of this universe should be organized hierarchically. Siblings are not assumed to be mutually exclusive, but a child must be a subset of its parent. For example,



The universe as described in the scheme is referenced by the question construct (use of a question within an instrument), variables, and NCubes. Describing these in a single location provides information on hierarchical relationships, and clear comparability of the universe of two questions or variables that reference the same universe. In addition, a universe can be described in both human-readable and machine-readable formats that provide explicit information on the definition of the universe. Note that variables allow for multiple universe links. When more than one universe is referenced, the universe is defined as the common area. For example a Variable that references both the universe “Female” and the universe “Employed” would be defined as Females in Canada who have a status of Employed in the Civilian Labor Force (16 years of age or older).

3.4 Methodology

If known at this point, include information on Methodology found in the datacollection schema. Methodology has an extension base of VersionableType. Given the breadth of the content, careful use of VersionRationale can help the user to quickly identify material sections that were changed or added in the new version. Methodology includes DataCollectionMethodology, TimeMethod, SamplingProcedure, and DeviationFromSampleDesign. In addition, you can note any datacollection software you plan to use in Software. Note that all of these sections are repeatable and that you should note separate methods in separate repetitions. You may need to reference a specific method later in the study and you can only do this if you have listed and identified each as a separate description. With the exception of Software, these are all IdentifiedStructuredStrings to allow for clear presentation of the material.

3.5 Collection Event

Each collection event should have its own entry. A collection event is normally described as the event (covering one or more days) that results in a raw data set which may or may not be linked to other raw datasets over time (through repetition) or from other sources for example a Parent survey linked to a Child survey. Each CollectionEvent provides a reference to who collected the data (DataCollectorOrganizationReference), a description of the DataSource, DataCollectionDate, DataCollectionFrequency, ModeOfCollection, CollectionSituation, and ActionToMinimizeLosses. While all of these except DataCollectionDate are repeatable within CollectionEvent, if the details of multiple CollectionEvents are listed within a single CollectionEvent section, the relationships between specific elements of information such as DataSource and ModeOfCollection can be lost.

Element	**Description of content**
DataCollectorOrganizationReference	References to the organizations or individuals responsible for collecting data. References are to the descriptions contained in an Organization Scheme.
DataSource	Provides a SourceDescription (structured strin), a SourceType (may be from a controlled vocabulary), the Origin which is a citation or URI to an external non-DDI description of the data, and Characteristic such as the level of documentation available for the source data or other factors like water damage that could affect the quality of the data source.
DataCollectionDate	Standard date type allowing a single date, range, or duration.

DataCollectionFrequency	This is the intended frequency of data collection, where the current collection event lies, or is meant to lie within a series. It consists of both a standard date type and the ability to add a code or string such as monthly, yearly etc.
ModeOfCollection	IdentifiedStructuredString
CollectionSituation	IdentifiedStructuredString
ActionToMinimizeLosses	IdentifiedStructuredString

4.0 Archives, Organizations, and Life Cycle Events

The Archive in the context of DDI 3.0 is the individual or organization responsible for the DDI instance at a given point in time. At the onset of a study this may be the principal investigator or their organization. The Archive module contains information that is both persistent (travels with the metadata throughout its lifetime) or is specific to the archive itself (information that has no relevance once it is distributed to another archive). At the onset of a study, the primary pieces of information entered in the Archive module include the identification of organizations, a reference to the current archives as listed in organization, and the addition of specific life cycle events as they occur. The contents of the major sections are provided below for context.

4.1 Archive Specific Information

The first item is a reference to the current archive as listed in Organization. Following this is a listing of the individual items that make up the instance as viewed by the archive. This can include associated data files, referenced schemes, etc. Each item in the list can have the following individual distinct pieces of information. Alternatively, multiple items can be treated as a Collection of Items or subordinate Collections. Flexibility was provided to support the common uses by different archives at different stages of the life cycle. DefaultAccess and FundingInformation for the specific Archive can be listed here and will apply to the contents of the items or collections listed in this section. For example this could include access being restricted to Faculty, Staff, and Students of an academic institution. The contents of both Item and Collection are listed below. Items that are unique to each are in italics to help the user determine the most appropriate usage for their particular archive.

Item content	**Use**
LocationInArchive	The location of an item within an archive, for example, if the archive needed to differentiate between multiple systems or different physical location options within the archive. For example a publicly accessible site or an off-line storage site.

Call Number	The specific identifier within this archive.
URI	The URN or URI of the item.
Format	*The format of the data or metadata holding.*
Media	*The current media of the holding.*
Study Class	Allows for an archive specific designation of a study class.
Access	Access information specific to this item
Original Archive Organization Reference	A reference to the source archive of the documented material. This is repeatable so that a chain of ownership can be recorded.
Availability Status	A human-readable description of availability.
Data File Quantity	Number of data files in the documented holding.
Collection Completion	A statement regarding the completeness of the documented holding.
Item	A subordinate Item

Collection content	**Use**
LocationInArchive	The location of an item within an archive, for example, if the archive needed to differentiate between multiple systems or different physical location options within the archive. For example a publicly accessible site or an off-line storage site.
Call Number	The specific identifier within this archive.
URI	The URN or URI of the item.
ItemQuantity	*The number of items in the collection.*
Study Class	Allows for an archive specific designation of a study class.
DefaultAccess	*Access information specific to the contents of this collection.*
Original Archive Organization Reference	A reference to the source archive of the documented material. This is repeatable so that a chain of ownership can be recorded.
Availability Status	A human-readable description of availability.
Data File Quantity	Number of data files in the documented holding.
Collection Completion	A statement regarding the completeness of the documented holding.
Item	A subordinate Item
Collection	*A subordinate Collection*

Note that access and access restrictions are provided at a variety of levels. This includes general access restrictions for the archive as well as specific access restrictions to all or parts of the data. Conditions for use, disclaimers, citation and deposit requirements and all related access forms are located here. DefaultAccess under ArchiveSpecific would apply to the contents of the specified archive. DefaultAccess at the Collection level applies to all the items within the collection whereas item specific information is found in Item/Access.

Element	**Use**
Confidentiality Statement	A structured statement
Access Permission	A form type including a statement of the access permission required, the form number and location, and an indication that the form is or is not required.
Restrictions	A structured statement
Citation Requirement	A structured statement
Deposit Requirement	A structured statement
Access Conditions	A structured statement
Disclaimer	A structured statement
Access Restriction Date	A standard date type allowing for a specified date or range.
Contact Organization Reference	A reference to and individual or office within an organization or to the organization itself.

Finally, there is funding information. In this location it is limited to information on funding sources specific to the archive, for example, if an archive received funding to collect, process, and provide access to a specific set of studies.

4.2 Organization

The OrganizationScheme is comprised of two basic structures: organization and individual. Each basic structure is made up of a description of the organization or individual and its components, the role of the organization/individual as related to the life cycle of the data, and relationship to other organizations or individuals.

The organization description provides each organization with an ID that can be referenced by other locations in the DDI instance. The name and abbreviation (or nickname) of the organization, along with contact information and member individuals should be provided here. An organization can belong to a group of Organizations directly as a child of the group or through inclusion by reference. Organizations can be related to other organizations in any way that can be described by the user of the DDI. Individuals can belong to organizations (as well as have organizations attached to the individual’s record internally). They carry their own contact information as well as information on language capabilities.

Designed to provide a flexible structure to describe organizations and individuals and their relationships to each other, the organization module captures information on organizations and individuals related to the study in a uniform way and allows the DDI document to reference these entries in relation to their specific roles within the life cycle of the study. The advantage, in addition to reducing redundancy, is the ability to capture the relationships between and within organizations, thereby clarifying the relationships of individuals and organizations related to the study.

Minimal entry for beginning a study would be the identification of the organization proposing to do the study:

```
<a:OrganizationScheme isMaintainable=”true” id=”MAINORG” urn=”
        urn:ddi:3_0:OrganizationScheme=MPC.MAINORG[1.0]”>
          <a:Organization isVersionable=”true” id=”MPC” urn=”
            urn:ddi:3_0:OrganizationScheme.Organization=MPC:MAINORG[1.0].MPC[1.0] >
                    <a:OrganizationName>Minnesota Population Center</a:OrganizationName>
                    <a:Nickname>MPC</a:Nickname>
                    <a:URL>pop.umn.edu</a:URL>
          </a:Organization>
</a:OrganizationScheme>
```

Any organization that is producing or managing a number of DDI documents should consider maintaining a separate OrganizationScheme within a published ResourcePackage. This makes tracking and updating organization changes easier and provides a clear historical trail. Note that Organization and Individual have to places to enter a number of specific Location information items including Telephone, URL, Email, InstantMessaging, and RegionalCoverage. Two approaches are possible. First, use the items directly under Organization to provide the current information and place Address, Country, and GeographicLocation in a Location element with a type of “current”. Move all of the current information (from both Location and elements under Organization to a Location element with a type “superceded” when location information changes. This could also include a note providing its valid coverage dates. The organization entry would have a version update and the VersionRational would state that there were changes in the contact information.

Organization name changes can be handled in two ways. First if the organization is essentially the same simply with a new name and you wish to retain the ID. To retain the original ID, version the Organization entry, moving the old OrganizationName to Nickname and entering the new organization name along with a VersionRationale comment. If the organization has changed, for example merged or changed mission, you may wish to provide a new entry with a new ID. You can then use Relation to link it to the old ID and note the effective dates of the old and new organizations as well as descriptive and/or keyword classifications of the relationship.

Careful consideration of both the information you wish to retain and how it will be maintained will help you determine how you may want to control the use of options within OrganizationScheme. If this involves disallowing specific elements, making an element required, or providing it with a default value, the archive is advised to create a DDIProfile detailing this specialized usage. In addition, the archive may wish to develop controlled vocabularies to use in managing the information held in this scheme.

Entries for Individual are structured in much the same way as Organization. Individuals and Organizations can be nested as desired. Users may wish to decide to focus on either nesting or relation references as a primary structure for internal relationship definitions.

4.3 Life Cycle Information

Life cycle Information is a simple listing of important events in the life cycle of the study or group of studies. Each event uses an IdentifiableType as an extension base and includes an EventType, a Date (this can be a simple date or a range), a reference to the organization or individual responsible for the event, and a description of the event. EventType is a CodeValueType which allows an organization to define specific codes relevant to their organization. DDI may publish a basic list of lifecycle events to provide a common list for DDI users. Life cycle events should be entered as they occur.

```
<r:LifecycleInformation>
<r:LifecycleEvent isIdentfiable=”true” id=”LC_1”>
        <r:EventType codeListID=”DDI_Events_List” codeListAgency=”DDI Alliance” >
        StudyDesign
        </r:EventType>
        <r:Date><r:SimpleDate>2007-04-01</r:SimpleDate></r:Date>
        <r:AgencyOrganizationReference>
        <r:Reference>
        <r:URN>
        urn:ddi:3_0:OrganizationScheme.Organization=MPC:MAINORG[1.0].MPC[1.0]
        </r:URN>
        </r:Reference>
        </r:AgencyOrganizationReference>
        <r:Description>
        Initial study proposal outlined and documentation  started.
        </r:Description>
</r:LifecycleEvent>
</r:LifecycleInformation>
```

5.0 Building and Documenting a Questionnaire

Data collection contains the major components of the questionnaire’s content and structure. In essence, a questionnaire contains questions and response options in an organized arrangement generally with instructions for the person answering either directly or through the interpretation of an interviewer. DDI 3.0 distinguishes all of these parts primarily to support reuse through category schemes, coding schemes, and question banks. Separating these component parts allows for a generic description of instrument flow that can be captured from computer-aided survey systems or used to instruct such a system.

To enter information into this section of Data Collection, you need to think about a survey as a collection of its component parts. Some parts are persistent in relationship to the content of the question and others are related to the use of a question in an instrument. The following chart differentiates these major divisions. These are entered and assembled in life cycle order, selecting the questions needed to measure the desired concepts, determining the allowable responses, setting up question order, and adding related text to guide the respondent through the questionnaire and inform capture of the raw data.

Persistent Content of the Question	**Use of the Question in an Instrument**
Question text • Simple text • Dynamic text	Order and Routing • Sequence • Loops • Skip patterns
Multiple part question [a question whose comparability is based on a particular question sub-series]	Universe
Response Domain • Open • Set categories • Special types (date, time, etc.)	Analysis Unit
Definitional text [specific to the question as opposed to its use in a particular instrument]	Instructions • Enumerator • Respondent • Coding (capturing raw data)

This type of deconstruction is needed to allow for support of question banks that both reuse question text and associate multiple response domains with the same question.

Questions are structured to support capabilities for dynamic text, multi-language comparability, and alternative response domains. The response to the same question can often be captured in multiple ways, for example:

Question: How old are you?
Response Domain 1:
  Numeric, maximum 3 characters
Response Domain 2:
  Under 18 years
  18 to 64 years
  65 Years and older
Response Domain 3:
  Under 5 years
  5 to 9 years
  10 to 14 years
  …continuing in 5 year cohorts

In addition, separating the flow logic, interviewer instructions, and identification of external materials used in the questioning process, allows DDI 3.0 to support a wide number of instrument types by providing generic instructions that can be interpreted by the individual instruments for presentation in multiple media formats.

Identifying these parts when looking at questionnaire can be more difficult.

```
IF LongIll=Yes THEN
    FOR i=1 to 6 DO
        IF (i=1) OR (More[i-1]=Yes) THEN
            Records up to six long-standing illnesses
            (IllsM[i])
            What (else) is the matter with you?
            INTERVIEWER: RECORD FULLY. PROBE FOR DETAIL. IF
            MORE THAN ONE MENTIONED, ENTER ONE HERE
            ONLY.

            Text: Maximum 60 characters
```

This question starts with question routing information that is specific to the instrument. It would use the IfThenElse construct as its initial organizing structure. This is followed by a Loop construct (or possibly a repeat while) that repeats the question up to 6 times. The question itself is found in another IfThenElse which prompts the inclusion of the dynamic text (else) in the question text. The text “*Records up to six long-standing illnesses”* appears to be the purpose of the question, but is actually the reason for looping the question within this questionnaire. As such it should be included in control construct as an InteviewerInstructionRefereence. The text “INTERVIEWER: RECORD FULLY…” would also be an InterviewerInstruction although it may be considered part of the question or question construct rather than as part of the IfThenElse or Loop construct. The question text is dynamic;

```
 <QuestionText>
        <LiteralText>What</LiteralText>
        <ConditionalText>
        <Expression>IF i>1 THEN: else</Expression>
        </CondidtionalText>
        <LiteralText>is the matter with you?</LiteralText>
</QuestionText>
```

“Text: Maximum 60 characters” provides information on the type of ResponseDomain (text) and the maximumLength (60) of the response.

Another common occurrence in printed questionnaires is the inclusion of routing information next to response categories.

H10a TOILET FACILITIES: What type of toilet is used by the household?

1. W.C.
2. Pit Latrine
3. KVIP
4. Bucket/Pan
5. Toilet in another house (different house) (Go to H11)
6. Public Toilet (WC, KVIP, Pit, Pan etc.) (Go to H11)
7. No facilities (bush/beach/field) (Go to H11)
8. Other (specify) ________________________________

In the above question there is both routing directions IF H10a > 4 AND H10a < 8 THEN H11 ELSE H10b. Also category 8 requires a write in response. In coded responses this would involve the use of another IfThenElse requesting a text response to specify the Other category using a separate question.

Occasionally complex responses are used including a check box, fill-in number, and categories. For example:

48b how much is your regular monthly payment on all second or junior mortgages and all home equity loans on THIS property?

Monthly amount—Dollars
$_ _ , _ _ _ .00

OR
_ No regular payment required

This is an example of a StructuredMixedResponse that contains:

```
<ResponseText><LiteralText>Monthly amount—Dollars</LiteralText></ResponseText>
<NumericDomain>…</NumericDomain>
<ResponseText><LiteralText>OR</LiteralText></ResponseText>
<CategoryDomain>…</CategoryDomain>
```

5.1 Question Construction

A basic question is an extention of VersionableType and contains a question text, question intent, response domain, concept references, and related visual aids. The presence of question intent allows you to use this structure for question development by capturing the intent of the question. This is also helpful when collecting information used as an indicator or when dynamic text changes or language differences result in varied wording of the question itself. The question text contains a Text element that serves as a container for a variety of text types and provides for a description and content. Note that questions do not contain display information for the content as this is specific to its use in an instrument.

Text types can currently be:

Text Type	Usage
Literal Text	The value is a static text string.
Conditional Text	Provides a condition on which the associated text varies (for example, gender).

All questions have a designated response domain. This may be a reference to previously defined category schemes or coding schemes using the Response Domain Reference, or by defining the response domain within the question.

Response Domain Type	Usage
Text	Text content is used for open ended, non-numeric, or mixed response. They may be constrained by acceptable content range and/or by length.
Date Time	Structures a specified data and/or time content
Numeric	Provides a numeric type code, scale, decimal positions, start and end value of accepted range, and the allowed interval.
Code	References a defined CodeSheme. CodeSchemes are used by both questions and variables although the same scheme may not be used by a question and its resulting variable due to recoding.
Category	References a defined CategoryScheme. A category scheme is a list of valid responses that do not have code representations assigned to them.
Geographic	A structure for capturing required information from GPS systems and other means of geographic location definition used in a data collection instrument.
StructuredMixed	Allows for use of multiple response domain types plus DynamicText in order to structure complex response domains that contain multiple response types.

CodeSchemes and CategorySchemes are defined in the logicalproduct and are used to describe the response domains of both questions and variables. In general, a category scheme is organized and assigned representational codes by a code scheme prior to its use in a variable such as code 0 equals Male and code 1 equals Female. A question may or may not use a coded category to capture a response. Coding is assigned when transferring a response to the raw data file.

Category schemes are used 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. This is most common in paper-based questionnaires, for example:

Question:
What is your marital status?
Response Domain:

○ Married
○ Single, never married
○ Widowed
○ Divorced

Question Schemes define a set of questions used in a study. They may include inline descriptions and/or references to external questions from question banks.

MultipleQuestions are those that require explicit responses to a set of subquestions. For example:

Q10. Read through the following list of candidates and for each candidate indicate how familiar you are with his position on free trade.

	1	2	3	4
Bill Clinton	○	○	○	○
Bob Dole	○	○	○	○
Ross Perot	○	○	○	○

Where 1 = Very familiar
2 = Somewhat familiar
3 = Not familiar
4 = Do not know of this candidate

The sub-questions consist of each candidate’s name as the question text and the CodeDomain.

Sequencing of response categories, subquestions of a multiple question, and questions within a series can be defined using an element provided in QuestionConstruct (ResponseSequence), MultipleQuestion (SubQuestionSequence), and Seqence (ConstructSequence). The user can designate the default of InOrderOfAppearance, Random, Rotate (rotates through a sequence), Other (defined in AlternateSequence).

5.2 Control Constructs and Instrument

The purpose of the DDI instrument element is to record the flow of a questionnaire, its use of questions, and additional component parts. Instrument is a maintainable object and has a MaintainableType. The documenter can define instrument types and provide a type code for the instrument as a whole. Software used to collect data is identified here along with a reference to other objects associated with the instrument such as a physical or image copy of the questionnaire. Other than this general instrument classification information, Instrument is composed of a series of Control Constructs nested inside a single master Control Construct. Generally the top level Control Construct would be Sequence, implying a start and end of the instrument as the Sequence tag is opened and closed.

Control Construct	**Use**
IfThenElse	Provides a condition to trigger the Then clause and alternative Else action. Then and Else are both Nested Constructs.
RepeatUntil	Provides an Until Condition and a construct that continues until the condition is met.
RepeatWhile	Provides an While Condition and a construct that continues while the condition is true.
Loop	Identifies the Loop Variable, sets its initial value, the condition that must be met to end the loop, and step (increment) value for the loop variable and the control construct that is to continue until the end of the loop is met.
Sequence	Defines a sequence of control constructs reflecting the flow of the instrument or a subsection of an instrument.
ComputationItem	Used to assign a code to a variable.
StatementItem	Statement type provides for the inclusion of additional text such as pretext or posttext.
QuestionConstruct	Provides for the insertion of a question into the instrument. In addition to the question content, the sequence includes ability to designate a response unit, analysis unit, and to reference a universe.

A ControlConstructScheme lists all the constructs found within the instrument as individual items. Constructs that define the organization of other constructs such as Sequence or IfThenElse, Loop, etc. do not include their subordinate constructs inline but by reference. The easiest approach is to make a list of all Questionconstructs, StatementItems, and ComputationItems so that they are ready to include by reference. Then review your flow logic, identifying routing patterns, sequences of QuestionConstructs and/or StatementItems and begin defining these from the lowest level of the nesting out until you have a construct (generally a sequence) that provides the start and end point for the full instrument. This is the ControlConstruct that the Instrument will point to.

For example if you had a questionnaire with a single skip pattern, 6 questions, and 2 statements you might have the following contents in your ControlConstructScheme:

Question 1

```
Question 2    if 4 GO TO Statement 2, Question 4
              Else GO TO Statement 1, Question 3, Question 4
Question 5
Question 6
```

```
<ControlConstructScheme isMaintainable=”true” id=”CC1”>
        <QuestionConstruct isVersionable=”true” id=”QC1”>…</>
        <QuestionConstruct isVersionable=”true” id=”QC2”>…</>
        <QuestionConstruct isVersionable=”true” id=”QC3”>…</>
        <QuestionConstruct isVersionable=”true” id=”QC4”>…</>
        <QuestionConstruct isVersionable=”true” id=”QC5”>…</>
        <QuestionConstruct isVersionable=”true” id=”QC6”>…</>
        <StatementItem isVersionable=”true” id=”SI1”>…</>
        <StatementItem isVersionable=”true” id=”SI2”>…</>
        <Sequence isVersionable=”true” id=”SQ3”>
                <ControlConstructReference isReference=”true”>
                        <ID>SI2</ID>
                </ControlConstructReference>
                <ControlConstructReference isReference=”true”>
                        <ID>QC4</ID>
                </ControlConstructReference>
        </Sequence>
        <IfThenElse isVersionable =”true” id=”IF1”>
                <IfCondition><Code>QC2 = 4</Code></IfCondition>
                <ThenConstructReference isReference=”true”>
                        <ID>SQ3</ID>
                </ThenConstructReference>
                <ElseConstructReference isReference=”true”>
                        <ID>SQ2</ID>
                </ElseConstructReference>
        </IfThenElse>
        <Sequence isVersionable=”true” id=”SQ2”>
                <ControlConstructReference isReference=”true”>
                        <ID>SI1</ID>
                </ControlConstructReference>
                <ControlConstructReference isReference=”true”>
                        <ID>QC3</ID>
                </ControlConstructReference>
                <ControlConstructReference isReference=”true”>
                        <ID>QC4</ID>
                </ControlConstructReference>
        </Sequence>
        <Sequence isVersionable=”true” id=”SQ1”>
                <ControlConstructReference isReference=”true”>
                        <ID>QC1</ID>
                </ControlConstructReference>
```

```
		<ControlConstructReference isReference=”true”>
			<ID>QC2</ID>
		</ControlConstructReference>
		<ControlConstructReference isReference=”true”>
			<ID>IF1</ID>
		</ControlConstructReference>
		<ControlConstructReference isReference=”true”>
			<ID>QC5</ID>
		</ControlConstructReference>
		<ControlConstructReference isReference=”true”>
			<ID>QC6</ID>
		</ControlConstructReference>
	</Sequence>
</ControlConstructScheme>
```

6.0 Data Processing

Data processing takes place at various points in the life cycle. The first such point concerns instructions for translating the response to the question into the raw data file. This is normally a direct capture. However, in the case of paper questionnaires this may include attaching codes to response categories or incorporating information not collected from the questionnaire. This could be geographic information, identifiers, recoded, or derived items. In addition, general instructions may address the handling of non-response, illegal multiple responses, or other common coding and processing issues.

The specific areas of information captured in DataProcessing include control operations, cleaning operations, weighting factors, data appraisal, and coding. Both ControlOperation and CleaningOperation contain a repeatable description field and an AgencyOrganizationReference to an organization or individual described in Organization. When entering this information separate out different operations or different steps. ControlOperation and CleaningOperations as well as the Description elements within them are unbounded. While these currently are not identifiable elements, keeping discrete activities separate is in line with the general philosophy of DDI 3.0.

Weighting factors are identified structured strings allowing for specific weight factors to be listed discretely and referenced. Although this allows for structured statements, consider separating descriptions of weighting factors (which may be structured) from any standard weights that are used as factors in category or variable statistics. You will want to refer to this as a number that was used in calculation.

Data Appraisal Information provides response rate (string), sampling error (structured string), and OtherAppraisalProcess (structured string). Once again, all of these are repeatable, and maintaining clear and discrete information on processes is important.

6.1 Coding

There are two different types of Coding describing general processes that are applied to a wide range of variables and specific generation instructions. All Coding elements are an extension of IdentifiableType so they can be referenced by ControlConstructs and Variables. Coding contains either GeneralInstruction or GenerationInstruction.

A general instruction pertains to coding operations such as the uniform handling of non-response or general imputation instructions. It contains a required repeatable Description and Command structure. The first is a structured string and intended to be human-readable. Command provides for a human-readable CommandText, an optional repeatable reference to a CommandFile, and an optional StructuredCommand that allows for inserting extentions to provide structured language for external namespaces such as MathML. When creating GeneralInstructions make sure that the relationship between the Description and Command is clear. Do not mix several Descriptions and/or Commands pertaining to multiple instructions in a single GeneralInstruction. Create a separate GeneralInstruction for each. GeneralInstruction also contains an element and attribute pair IsOverride and @isOverride. When @isOverride is changed from its default value of “false” to “true”, the element IsOverride must be completed. This element provides the ID of the GeneralInstruction that is being overridden.

GenerationInstruction is more complex as it deals with specific recodes and derivation instructions. An attribute, isDerived, indicates whether this is a simple recode or a more complex derivation instruction. GenerationInstruction In addition to the Description and Command structures found in GeneralInstruction it provides identification of Questions, Variables, and ControlConstructs used in the generation command. You are able to assign a mnemonic to the Question or Variable for use in the command equation. It also contains a special Aggregation description which provides the aggregation method and identifies the Independent and Dependent variables required for the computation. This structure is also available in CodeMap for use in defining recodes from the Source to the Target structure.

7.0 Creating a Basic Data Dictionary

A basic data dictionary in its simplest form consists of a description of variable contents, information on the universe, the concept represented by the variable, the measurement unit, analysis unit, the number of respondents, and the location of the data field in the physical data store. DDI 3.0 separates these parts into separate category schemes, code schemes, variable descriptions, and physical location.

7.1 Category Schemes

Category schemes should be started early in the process. Category schemes are used by questions as response domains and by code schemes in preparation for use in variables. A category scheme can include a set of related categories or include all the categories used in the study. Note that category schemes are maintainable and you should consider creating separate category schemes for subsets of categories that may be reused in other studies. This is especially true if your questionnaire uses un-coded categories. One group of categories you may wish to construct is one including non-response categories used in the study. Any category used in the study should only be listed once. Reuse of a category in multiple code schemes implies comparability.

As a maintainable object, Category Schemes are an extention of a MaintainableType and contain descriptive information about the scheme (label and description) to provide information on what the scheme contains. Its content is made up of a set of category descriptions which may or may not be organized into category groups. A category is an extension of VersionableType to allow for updates and changes over time, such as changes in a label or description. The label is repeatable to handle multiple languages. This is the actual category term as used in the question response domain or variable representation. The description is repeatable to handle multiple languages. When determining the comparability of categories, base the comparison on the description not on the label. For example, the category “Chemist” in British occupation codes has the same description and is comparable to the American occupation “Pharmacist”.. In addition, the object Generation allows for description of the command used to generate the contents of this category, for example “Other Income” equaling total income minus wage/salary income.

7.2 Code Schemes

Code schemes organize categories from one or more CategorySchemes and provide the code representation for the category as it is found in the question or variable. A code scheme may be flat or hierarchical and the hierarchy can be regular (each branch having the same number of levels) or irregular (the number of levels varies by branch). A code within a code scheme is identified by its unique code value. A level or specific code may or may not have data associated with it. A code without associated data acts as a category group as found in earlier DDI structures. Its sole purpose is to group a subset of categories/codes and provide a grouping label through reference to a category.. A CodeScheme contains only a single code scheme and should provide all legitimate categories. If a standard codeScheme is used for non-response, a single CodeScheme can describe that and be included in each relevant codeScheme by reference.

To build a CodeScheme you must have completed the relevant CategorySchemes. A CodeScheme has a MaintainableType, Label, and Description. It may reference one or more existing CodeSchemes. For example, if there is an existing CodeScheme for Gender (0 = Male, 1 = Female) and another for Non-response (9 = No response), a single CodeScheme could be created to include these three valid responses, 0 = Male, 1 = Female, and 9 = No response. If all or a majority of the categories come from a single CategoryScheme, it can be declared once, thereby creating a default value for the maintainable object of category references allowing subsequent category reference to list only the category's ID and Version information. This can be overridden by providing a complete URN at the category reference level. If the CodeScheme is hierarchical, you must indicate if it is regular or irregular. Flat CodeSchemes are the default and do not require this element. Hierarchical CodeSchemes must have their levels described so that they can be referenced by Variables using a limited number of levels and to understand the relationship between the levels and their contents.

A level has a name, description, relationship type, and interval. The relationship type describes the relationship of the categories contained by the level. They can be nominal (no implied order), ordinal (ordered as presented in the description), interval (both ordered and with a consistent interval between categories), ratio (ordered and with a consistent interval ratio), or continuous (either interval or ratio, use when unsure of the interval type) . If the categories have an interval relationship, provide the anchor (base value for the first category) and increment value of the interval. This information is provided for each level as this may vary by level, for instance, a single hierarchy may have ordinal or interval relationships at upper levels and nominal at the lowest level.

Codes are then listed containing a reference to the category being represented and a code value, plus a nested code to allow for building hierarchies. A code has two attributes, a level number (optional) to indicate the level of the coded category, and isDiscrete. The field isDiscrete has a default value of "true". Set this attribute to "false" if it has a subordinate level.

7.3 Describing Variables

A variable is part of a variable scheme and is made up of the following components:

Element	**Usage**
VersionableType extension base	This is how the variable is referenced by other objects in the instance
Name	Contains an optional Name for the variable such as a mnemonic. This is available in all identifiable elements but is most commonly used in Variable. Note that it is repeatable for language and geographic differences.
Label	This is a short human-readable label description of the variable.
Variable Definition	Additional textual description of the variable

	which may be used to provide extended detail concerning the variable.
Universe Reference	Universe is described in ConceptualComponent. This allows the universe of the variable to be seen in context of the full universe structure and provides comparability for two variables using the same universe. A universe is assumed to be the fullest universe of the study if it is not provided but given the complexity of multiple data products being produced from a single study, it is recommended that a universe be explicitly identified. Variables used as component parts (attributes or dimensions) of NCubes do not require a universe as this is defined in the NCube.
Concept Reference	Similar to Universe in terms of concepts being described within the ConceptualComponent. All variables must have one and only one concept declared.
Question Reference	References to all questions as expressed in an instrument used to determine the value of this variable.
Embargo Reference	References access restrictions for this variable.
Response Unit	Who provided the information for this variable. This may be the same as the response unit of the question as used in the instrument.
Analysis Unit	Describes who or what this variable is an attribute of (who or what is described).
isTemporal isGeographic isWeight	Three Boolean attributes all with the default value of “false”. If true, reset the value of appropriate attribute to “true”.
Representation: With optional attributes of measurementUnit, aggregationMethod, additivity	Describes how the variable is represented in the data file according to one of the following. Attributes are optional as this information is listed in the NCube when the variable is used as a dimension or attribute of an NCube.
WeightVariableReference	References the variable to be used with this variable.
StandardWeightReference	References the weight factor described in Data Processing.
ConcatenatedValue	Describes 2 or more variables, that when concatenated provide the value of this variable. Used primarily to create a virtual variable such as a multipart link whose parts are described by separate variables.

CodingInstructionReference	References the specific coding instruction in Coding used to create this variable. This may be a recode or derivation instruction.
ValueRepresentation	Provides the actual representation used by the variable. This is an abstract object that acts as head of a substitution group.

Some of these elements may have been compiled at earlier stages of the study. For example, the concept and universe structures, questions, a number of category and possibly code schemes, and some of the recoding or derivation process will already exist. The first step is to complete those component parts.

The Value Representations available are described below. Note that CategorySchemes and CodeSchemes should be created first. CodeSchemes should describe the full structure of the code. A variable can use all or parts of the CodeScheme thereby retaining the relationships among the various parts or levels of the structure. Each of these types extend the contents of VariableRepresentation noted in the above table.

7.3.1 Text

TextRepresentationType allows you to specify the minimum and maximum length of the text content and to provide a regular expression that can be used to restrict the content of the text string. For example, a United States ZIP Code is a text string because the leading zeros carry significance. As a text string a five digit ZIP Code would have a minLength of 5, a maxLength of 5, and a regular expression of [0-9]*. Please note that text letters that represent categories labels are described with CodeScheme. The value of a code may contain any character.

7.3.2 Date/Time

DateTimeRepresentation requires the use of one of the following types corresponding to the W3C XML schema xs: datatype.

Code	**W3C XML schema xs datatype**
DateTime	[xs:dateTime] Contains both the date and time as <date>T<time>
Date	[xs:date] Contains the full date from the Gregorian calender YYYY-MM-DD unless an alternative format is provided
Time	[xs: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	[xs:gYear] Contains the 4 digit year YYYY

Month	[xs:gMonth] Contains the 2 digit month MM
Day	[xs:gDay] Contains the 2 digit day DD
MonthDay	[xs:gMonthDay] Contains the 2 digit month followed by the 2 digit day as MM-DD unless an alternative format is provided
YearMonth	[xs:gYearMonth] Contains the 4 digit year followed by the 2 digit month as YYYY-MM unless an alternative format is provided
Duration	[xs: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".

An optional format element can define an alternative format for the data field such as MM-DD-YY. The default is the W3C format.

7.3.3 Numeric

Note that numeric should not be used for numbers that represent categories (1 = Male, 2 = Female). The use of a numeric representation suggests a form of count (6 years, 20000 Euros, 2 children). Numeric representation provides a start and end value for the range, scale, the number of decimal positions, interval, and a numeric type code. The type code is required and is represented by one of the following.

Code	**W3C XML schema xs datatype**
BigInteger	[xs:integer]An integer of unlimited size. An integer datatype

	corresponding to W3C XML Schema's xs:integer datatype.
Integer	]xs:int] An integer number can hold a whole number, but no fraction. Integers may be either signed (allowing negative values) or unsigned (nonnegative values only). An integer datatype corresponding to W3C XML Schema's xs:int datatype.
Long	]xs: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). A numeric datatype corresponding to W3C XML Schema's xs:long datatype.
Short	]xs: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), A numeric datatype corresponding to W3C XML Schema's xs:short datatype.
Decimal	]xs:decimal] A real number (allows fractions expressed as decimals). A numeric datatype corresponding to W3C XML Schema's xs:decimal datatype.
Float	]xs:float] Real numbers that may be stored in scientific notation (example: 20.0005, 99.9, -5000.12, 6.02e23). A numeric datatype corresponding to W3C XML Schema's xs:float datatype.
Double	]xs:double] Float of up to 32 bits. A numeric datatype corresponding to W3C XML Schema's xs:double datatype.
Count	Ordinal number of objects in a finite set, discrete. A simple incrementing Integer type. The isSequence facet must be set to true, and the interval facet must be set to "1".
Incremental	A value that is continuous and infinite can be interval or ratio. This value indicates that the value increments according to the value provided in the interval facet, and has a true value for the isSequence facet.

7.3.4 Code

The value representation for code allows a single CodeScheme to be applied in a number of ways. You can define which portions of the CodeScheme are being used in a particular variable. The default is to include all codes described in the codeScheme. In the case of a hierarchical codeScheme, this means that the response can include all levels of the hierarchy. Alternatively, the following objects can be used to constrain what is included in the variable.

IncludeLevel	Identify specific levels to be included in the variable
IncludedCode	Reference to included codes
DataExistence	Include the most discrete items only (this allows inclusion of all codes designated as the most discrete which could be a level or in the case of an irregular hierarchy, the rightmost

	codes in each branch of the tree's hierarchy

Example:

Irregular Hierarchy
1 Metals (level 1)
2 Iron (level 2)
3 Bar Iron (level 3, isDiscrete="true")
4 Caste Iron (level 3, isDiscrete="true")
5 Copper (level 2, isDiscrete="true")
6 Non-Metals (level 1, isDiscrete="true")

Variable 1 – using IncludeLevel
Includes Level 1 and Level 2
Valid responses are 1, 2, 5, 6

Variable 2 – using IncludedCode
Includes valid responses 3, 4, 5

Variable 3 – using DataExistence
Includes valid responses 3, 4, 5, 6

Variable 4 – no constraints noted
Includes valid responses 1-6

7.4 Data Relationships

The Data Relationship portion of a logical product describes the logical records described in terms of their coverage, unique identifiers, and interrelationships. This section is meant to reflect the information often found in the "How to use this file" type sections of a traditional codebook. All data sets have one or more logical record types. Note that this is referring to the logical structure of the records not their physical layout which may be hierarchical, rectangular, or held in a relational data set. This section is focused on what variables are available to link data together and to assist the user in identifying unique cases with a record type. At a minimum, the human-readable description should be completed. Ideally the detailed information should be completed to support machine-actionable exploration of the data set.

The Data Relationship section contains three main parts. First is a human-readable description of the logical record types contained in the data set and the relationships among these record types. The second part describes each logical record type. The record type descriptions are Identifiable so that they can be referenced by the physical data structure.

The variable value reference provides the identification of the variable that differentiates one record type from another and the value of this variable for the record type being described. In files with a single record type this would not be used. However, most files with more than one record type have a variable such as RecordType that provides a value such as “H” for household record and “P” for person record. Older data files may have an identifier for the first record with subsequent records (records of another type nested within them) simply being those records that do not have this value. The attribute hasLocator must indicate whether or not the record has such a variable.

Support for multiple parts allows for the identification of any variable that provides information that can be use to identify which segment of a logical record you are dealing with. This is a common practice for long logical records that are being stored within the constraints of a particular storage structure (Excel files have a record length, old SPSS and SAS packages had a maximum record length that could be supported). Support for multiple records is often found in elements such as LogicalRecordPartNumber, which appears in the 1990 and 2000 U.S. Census Standard Summary Files. Other older files have record segments without support for multiple parts, and identification is based solely on record order within the file. This section provides both the variable used to identify a record part and the values available.

Case specification provides for one or more means of identifying a unique case within a record type. For example, a record within its original file structure may have a unique record ID variable, often used to link it to other records. However, there may also be other identifiers, either individual fields or combinations of fields, that allow for case identification. A common example of this is the geographic codes of aggregate data files. Case specification allows for alternative identification for various case types. For example, in a geographic case file, a County level record may require one set of variables (state and county codes) to identify it while a Place requires another set (state and place). Case specification allows for different sets of identifier variables to be declared for different values of a parent identifier.

The last part of the record description is a listing of the variables belonging to the logical record. This is a special type of Variable Group and should be declared within the Data Relationship rather than as a standard Variable Group.

The third section of Data Relationship deals with record relationships. These are declared in a pair-wise fashion to account for all supported relationships. This is done by defining the variable and value of the linking variable in the source record and its matching point in the target record. The relationship is defined through the VariableValueRelationshipType attribute as “parent”, “child”, or “sibling”. The value relationship by default is “equal”, but can be set to “GreaterThan”, “LessThan”, “GreaterThanOrEqual”, LessthanOrEqual”, or “NotEqual”. Multiple variable links should be handled by creating a variable with a concatenated value and using it as the link variable. This relationship is identifiable so that it can be referenced by the physical data product, reducing repetition of the information.

Note that if you are creating links between multiple logical product sections, you can put this information in a single logical product (defining all the relationships in the collection of logical products), or create a logical product that contains just the data relationship between all other logical products. This is useful when describing links between files in a longitudinal series, where there is a link between a person record occurring in one year and that person's record in another year or wave.

7.5 NCubes

NCubes are used to describe the data matrices created through crosstabulation and aggregation of microdata. An NCube represents a matrix where each cell intersects each dimension of the matrix in one and only one location. For example, AGE by SEX by COUNTRY OF RESIDENCE. A cell has an "address" which provides its intersect code for each dimension. In this case, "5,1,6" would be the fifth value of AGE, the first value of SEX, and the sixth value of COUNTRY OF RESIDENCE.

An NCube is an extention of a VersionableType, and contains Label, Description, a Universe Reference, ImputationReference, ResponseUnit, and AnalysisUnit. In this, it is very similar to a variable. An NCube must be constructed of one or more dimensions which are described as variables. The dimensions identify both a variable and a "rank" order so that the coordinate address will be clear. In the above example AGE has a rank of "1", SEX a rank of "2", and COUNTRY OF RESIDENCE a rank of "3". Additional attributes can be attached to all or part of the NCube. These could be cell level suppression flags described by a variable, footnotes or source notes, or whatever is required. By describing regions of the NCube with Coordinate Groups, one or more attributes can be attached to a cell, a dimension or dimension value, or a specific sub region of the NCube defined by the intersect values. A common use of the attribute within the NCube description is to identify cells that contain no data by definition. Many NCubes are created through the process of cross-tabulation and then collapsed when published to save storage or printing space. For example, the U.S. Census has a table of Number of persons per household by Household type where Household type contains Non-Family and Family Households. The number of person's categories run from 1 to 9 or more. This table is usually published as a one dimensional matrix Number of persons per household by household type.

LABELS	**Cell coordinates**
Nonfamily Household:	[no data used a category group label]
1	1
2	2

3	3
4	4
5	5
6	6
7	7
8	8
9 or more	9
Family Household:	[no data used a category group label]
2	10
3	11
4	12
5	13
6	14
7	15
8	16
9 or more	17

Note that 1 person Family Household is missing. By definition a “Family” is two or more persons so this cell would always be zero or have no data listed.

The layout of the NCube structure in DDI 3.0 allows for the following table structure and attributes description.

LABELS	Dimension 1:	
Dimension 2:	Nonfamily Household	Family Household
1	1,1	2,1
2	1,2	2,2
3	1,3	2,3
4	1,4	2,4
5	1,5	2,5
6	1,6	2,6
7	1,7	2,7
8	1,8	2,8
9 or more	1,9	2,9

The attribute would reference a Variable that declares it’s sole value as blank with the label “Definition as a Family Household requires a minimum of 2 people in the household”. The the value is attached at the CoordinateGroup which defines the CoordinateGroup as any cell with the dimension 1 value of 2 and dimension 2 value of 1 (Family Household, 1 Person).

Attribute values can be declared within the metadata if they are consistent for all instances of the NCube (such as footnotes or definitional values) or can be stored in the data file and attached to the cell in the RecordLayout definition.

The Measure of an NCube must be described by a variable in order to capture its universe and various aspects of the measure. For example, a measure of an NCube Persons Per Housing Unit may have a dimension denoting the number of persons per housing unit and the measure is a count of housing units falling into each definition. Alternatively, it may be the percentage of housing units with the designated number of people. In order to capture both the dimension and the measure, you need to have two unique definitions and their associated concepts. Some measures such as percentages require definition of the values that are used for the independent and dependent variables used to calculate the cell content. For example a percent could be the cell's percent of the universe or the cell percent of one of its dimensions. This can be handled one of two ways. First a separate variable can be created for each percentage or other aggregation using the specific variables to describe its calculation in GenerationInstruction. A variable pointing to this GenerationInstruction would be used for the measure. Alternatively a generic variable "Percentage of Housing Units" can be created with a GenerationInstruction that states that the value is calculated by dividing the dependent variable divided by the independent variable. The AggregationDefinition within Measure then declares which dimension(s) are used as the Independent variable and which are used for the Dependent variable. It can also note that the universe of the NCube is the independent variable. For example, when the NCube was Age by Sex and the Percent was provided for the each cell count divided by the total population of the NCube. If the percentage was the percent of 2 year olds who were female, Age would be the Dependent Variable and Sex would be the Independent variable.

Note that an NCube and a table are not synonymous. A table may consist of a number of NCubes that share a common dimension, such as AGE by COUNTY OF RESIDENCE followed by SEX by COUNTRY OF RESIDENCE, where there is no intercept between AGE and SEX (for example no "Males 5 to 9 years of age").

Table 1	Sex		Age		
	Male	Female	Under 18	18 to 64	65 or more
Region 1					
Region 2					
Region 3					
Region 4					

In Table 1 there are two NCubes Sex by Region and Age by Region. By definition the cells of an NCube must intersect each dimension of the NCube at one and only one point. None of the cells in the above table intersect both the Age dimension and the Sex dimension. They share a common dimension of Region and will each point to the same Variable to describe this dimension.

8.0 Physical Data Product

Physical Data Product was designed to describe the physical storage structure of data including the breakout of logical data records into multiple physical parts, a link to the logical record being stored, and the specific structure of each record. The general structure of this section is split into describing the Gross structure of the physical records (what logical record is contained and how physical records should be linked), and the details of the RecordLayout (this may include data inline for some structures). Note that any logical product may have more than one physical storage structure. Frequently, files are stored in multiple formats, for example, a fixed format archive copy and a database copy that supports a specific use of the data file. In turn, each physical record description can represent multiple physical files of data. These files of data may represent different record types (different RecordLayout structures) within the dataset or different subsets of records within the full dataset (multiple PhysicalInstances) such as a datafile containing just the records for the state of Arizona or just the records for females in the study.

Data can be stored in a wide and growing range of physical data structures. The common archival structures of fixed format and delimited files are described very much as they were in earlier versions of DDI. DDI 3.0 has tried to facilitate the management of the same data held in multiple storage structures and multiple datafiles. A single logical record of data may be storage as a single record string, broken into physical segments to accommodate line length limitations in some software, stored with all segments in the same file or different files (like relational data files), use a variety of proprietary and nonproprietary data structures, and be divided in multiple files of record subsets to facilitate processing. Within this, numerous pieces of information regarding each dataitem need to be recorded but much of it is repetitious and could be provided as default values. In order to handle this complexity, DDI 3.0 approaches the description of physical storage as follows:

Major Component	**Use**
Common Schema Components: • MaintainableType • OtherMaterial • Note	Users may choose to maintain a single PhysicalDataProduct which describes their holding. Whether representing a single StudyUnit or a whole collection, OtherMaterial related to physical processing, analysis tools, systems information or other related materials or notes can be stored here.
PhysicalStructureScheme: • MaintainableType • Description • Label	A separately maintainable scheme, PhysicalStructureScheme is basically a listing of individual PhysicalStructure descriptions included inline or through inclusion

• PhysicalStructureReference • PhysicalStructure	by reference of another published PhysicalStructureScheme
RecordLayoutScheme: • MaintainableType • Description • Label • RecordLayoutSchemeReference • BaseRecordLayout	A separately maintainable scheme, RecordLayoutScheme is basically a listing of individual BaseRecordLayout descriptions included inline or through inclusion by reference of another published RecordLayout Scheme. Note that BaseRecordLayout is an abstract for a substitution group including: • RecordLayout (ASCII fixed or delimited) • DataSet • NCube record layout • Tabular NCube record layout • Inline NCube record layout • Proprietary record layout

8.1 Physical Structure Scheme

PhysicalStructure is an extension of VersionableType and contains one or more LogicalProductReferences. Note that a LogicalRecord can contain parts of more than one LogicalProduct, this element allows a system to identify which LogicalProduct contents will be referenced from the physical record descriptions. It also contains one or more GrossRecordStructure descriptions. The remaining objects within PhysicalStructure are optional and provide default values that apply to all RecordLayouts referencing a PhysicalRecordSegement described within this PhysicalStructure unless overridden at a lower level of description. DDI 3.0 has provided default options a multiple levels and the user needs to determine how they wish to organize and manage their collection to take advantage of this feature. For example, if the Format (for example Fixed Format) is declared at this level, the PhysicalRecordSegments described here cannot be referenced by a ProprietaryRecordLayout or DataSet. However, it may be entirely appropriate to define the DefaultDecimalSeparator or DefaultMissingData here to avoid having to enter this information for each DataItem. Default declarations are also available all NCube storage descriptions as they can vary from NCube to NCube. These same elements are found in the PhysicalLocation of the DataItem description and can be described there. If defaults are used, the value in the PhysicalLocation will override the default value.

Element	**Usage**
DefaultDataType	Content is xs:string but preferred use is any W3C datatype
DefaultDelimiter	Allowed values are: Empty (default), Tab, Blank,

	AnyString. If a delimiter is used, free field (delimited data) is assumed; binary formats are not allowed. This is the delimiter between data items in a delimited file.
DefaultDecimalPositions	This is the default value of implied decimal positions (how many positions to the right of the decimal are included). For example, if the DecimalPosition is 2, then a DataItem value of 8 would be the equivalent of .08.
DefaultDecimalSeparator	There is no default value for this as a "." is common in the US and "," in Europe.
DefaultDigitGroupSeparator	A grouping separator separates groups of digits such as thousands from hundreds. Once again there is no default as "," is common in the US and "." Is common in Europe.
DefaultMissingData	Standardized handling of missing data across the dataset.

Gross Record Structure provides information on the gross or general physical structural of a logical record as it appears in a dataset. First is the LogicalRecordReference which links the physical description to the content information provided for the LogicalRecord as described in DataRelationship. An attribute, numberOfPhysicalSegments is set to a default value of one and should reflect the number of PhysicalRecordSegments defined for the record. You must describe at least one PhysicalRecordSegment (that is the full record) in order to provide the link to the RecordLayout. The minimum description for a logical record contained in one physical record is an ID (it is an extension of IdentifiableType), the attribute segmentOrder at its default value of "1" and attribute hasSegmentKey at the default value of "false".

A logical record that supports multiple segments may be stored as a single record, multiple segments in a hierarchical file, multiple segments in a data file per record segment, or a combination of these. Legacy data often has no structural support for multiple segments but is stored in segments in hierarchical files where the record order alone defines the segment. If the logical record has been separated into more than one physical segment the segmentOrder would be incremented and two optional elements may be used. If there is a segment Key (a variable that identifies the segement type, the attribute hasSegmentKey is changed to "true" and the element KeyVariableReference is provided giving the reference to the Key variable and a value of the variable for the specific segment.

Note that this is a reference is to a single variable so if this is a concatenated key, a variable whose content is the concatenation of two or more other variables must be described in the logical record and included in the LogicalRecord. In addition, a FileNameIdentification string can be provided which indicates how a segment may be located by its file name structure. For example, in 2000 the U.S. Census published Summary File 1 in 39 files for the US and for each state. The file name convention was xxnnnnn_uf1.zip where xx was either us or the state two letter postal abbreviation and nnnnn was the segment number.

8.2 Record Layout Scheme

RecordLayoutScheme is an extension of MaintainableType and contains a Label, Description and means of including one or more record layout structures inline or by referencing a publish RecordLayoutScheme. The element BaseRecordLayout is the abstract type for a collection of substitution types to describe the details of a record layout. This structure allows the development of RecordLayout descriptions that are specific to the needs of current and future storage systems, including proprietary systems like statistical software. BaseRecordLayout is an extension of IdentifiableType and contains a PhysicalStructureReference, CharacterSet (US ASCII, EBCDIC, UTF-8 etc.), and ArrayBase (1 or 0). If CharacterSet is unknown, say for a proprietary structure, enter “unspecified”. ArrayBase provides the assumed first position in any array used in describing this dataset. For example, codebooks historically assume the first character in a data record is “1”, that is, an array base of “1”. Unix systems and many programming languages assume an array base of “0”. If the codebook information uses an array base other than “0” the base level for any array handled by the system must be declared. If not, all start positions would be offset by one character. PhysicalStructureReference is a specially structured reference to a single PhysicalRecordSegment. It contains a reference to the PhysicalStructure and a separate element, PhysicalRecordSegmentUsed, that provides the ID of the PhysicalRecordSegment. Note that a BaseRecordLayout can only link to one PhysicalRecordSegment. If you have a logical record that is stored in some files as a single record and other files as multiple records, you must create two different PhysicalStructures to describe the physical stores of the logical record.

8.2.1 RecordLayout

The traditional archive format description of fixed format and delimited files (similar to earlier DDI versions) is described in RecordLayout as found in PhysicalDataProduct. All other BaseRecordLayout substitutions will be found in separate schemas. In addition to the elements inherited from BaseRecordLayout, RecordLayout allows for a DefaultVariableSchemeReference to limit the need to repeat this information for each DataItem’s VariableReference, an attribute namesOnFirstRow with a default value of “false”, and a list of one or more DataItem descriptions.

DataItem contains a VariableReference and PhysicalLocation. PhysicalLocation is used or extended by a number of other BaseRecordLayout substitutions. Its contents should be familiar to uses of earlier versions of DDI.

Element	**Use**
StorageFormat	Overrides DefaultFormat. Is a CodeValueType which allows for a controlled vocabulary
Delimiter	Overrides DefaultDelimiter
StartPosition	Used in fixed format files this is the position of the first character of the data item in the record
ArrayPosition	Used in delimited files. This is the array number of the data item. Note that the first item in the array should have a value corresponding to the ArrayBase declared in the RecordLayout.
EndPosition	The position of the last character of a data item in a fixed format file
Width	The actual width of a data item in a fixed format file or the maximum width in a delimited file.
DecimalPostions	Number of decimal places with an implied decimal separator. Another expression of the decimal scalling factor (SAS). Default value is “0”
DecimalSeparator	Character used to separate the integer from the faction portion of the number if it is used in the data set. Allowed values include: None (default), Dot, Comma, Other. Overrides DefaultDecimalSeparator
DigitGroupSeparator	Character used to separate the sections of an integer (between 100 and 1000 for example) if it is used in the data set. Allowed values include: None (default), Dot, Comma, Other. Overrides Default DigitGroupSeparator
LanguageOfData	Use two-character ISO Language Code to indicate the language of the data content. Applicable for text fields.
LocaleOfData	Use two-character ISO country code as a supplement to the LanguageOfDataCode

Note that when describing a fixed format file you must use either EndPosition or Width in conjunction with StartPostion. You may use both if desired. As you can see from the number of elements that override defaults that listing a default value that applies to only half the data items in the records, would considerably reduce the size and content of the metadata description. In general, it is best to provide the most common structures at the default level (for example DataType=”Integer”) and enter an override value for “String” or “Character” data items.

8.2.2 DataSet

DataSet allows for inline inclusion of data in the metadata file. This is valuable for small datasets and particularly for small statistical tables. This identifies the following items:

Element	**Usage**
IdentifyingVariable	References the variable containing the primary key or index value.
DefaultVariableScheme	By identifying the variable scheme here, one can enter just the ID for each individual item, reducing repetition.
CHOICE:	
RecordSet	Storage structure for a traditional rectangular data structure. Each array of data is for a single record with multiple variables.
ItemSet	Allows data items to be stored in a random order.
VariableSet	Storage structure that is transposed from the traditional record structure. Each array of data is for a single variable with a value for each record in record order.

The choice between three layouts provides flexibility to store data in whichever format is optimal for your particular use. A RecordSet requires a list of variables in order using VariableOrder (a simple list of VariableReference elements listed in order of appearance in the array. This is followed by Record (repeated for each record in the dataset) which is the array of values for all variables in the record. Note that for sparse arrays (arrays with missing values) a separator other than “blank” must be used in order to express the correct position in the array. For example:

For Var1=1   Var2=8 Var3=   Var4=10   Var5=
With Blank as delimiter: 1 8 10
With Comma as delimiter: 1,8,,10,

Clearly indicating the missing content for Variable 3 and 5.

ItemSet is a list of ItemValues, each with a VariableReference and a RecordReference (string containing the value of the IdentifyingVariable), and the value of the variable.

VariableSet contains a list of VariableItems each containing a VariableReference and a Value. The assumption is that each record will occupy the same position in each array and can be identified by the value provided in the VariableItem referencing IdentifyingVariable. As with RecordSet, datasets with sparse arrays will need to use a separator other than “blank”.

8.2.3 NCube Record Layout (Normal)

The first thing to note about NCube physical record layouts is that the RecordLayout can handle ONLY those dataitems found in NCubes. If your record has identifying variables fields in addition to the NCube content, you will need to describe a minimum of 2 PhysicalRecordSegments so that the appropriate RecordLayout can be used to describe each part. These will later be recombined in the PhysicalInstance which can contain multiple RecordLayout references stored as concatenated strings or as hierarchical structures.

The basic NCube structure is for storage structures where one or more NCubes are stored as records with the data items (cell data) strung out in sequence with or without a string of non-NCube variables that define the case. Census aggregate summary files are often structured in this way, where each record is a geographic area with the data for 100 or more tables strung out in a single logical record. This type of structure facilitates record subsets as it does not require a case identification as a dimension of the NCube. This structure can also be used for cases where all data items are described by the NCube. The NCube record layout provides a reference to a specific NCube, identification of any attributes provided for the NCube, and physical location of a data item in a record and association to the cell coordinates, attribute type, and measure. To identify the standard content of the NCube such as a count, provide the NCubeReference, use the Coordinate element to identify the coordinate number as described in the logical product, and the value of the coordinate or the variable where the value is being held. In addition you can provide similar location information on attributes associated with all or parts of the NCube through the use of attribute or coordinate group. Use coordinate group when the attribute is not associated with the full NCube.

In addition to the items inherited from BaseRecordLayout, the NCube RecordLayout is a simple series of NCubeInstance descriptions. Note that an NCube that is split between 2 or more physical records must be described in 2 separate RecordLayout descriptions. Any cell of an NCube for which there is no DataItem is assumed to be missing from the PhysicalRecordSegment being described. It may be missing because it has been declared as being “empty” by definition or because it is stored elsewhere in another PhysicalRecordSegment.

Each NCubeInstance has a reference to a single NCube description in a LogicalProduct. It may have an attribute attached at the NCube level for the NCube as a whole or a specified region of the NCube (as defined in LogicalRecord). The value of the Attribute may be declared in the metadata or, using the standard PhysicalLocation structure, reference the location of the attribute value in the dataset. For example, in many economic tables, suppression flags are attached to the full table or perhaps to a specific level of detail for a variable. In these cases a single flag applies to multiple cells in a table in a consistent fashion for each location. Some locations have suppressed data, others do not. Attribute at the NCubeInstance level is used only for these situations, when a single dataitem holds content that applies to multiple cells in the NCube matrix.

The remainder of the NCubeInstance content is the DataItem along with the ability to set the standard default values for the NCube as a whole. The DataItem of an NCube is associated with the NCube by its cell coordinates, the point of intersection on each of the dimensions of the NCube listed in rank order. The cell coordinates are provided by the repletion of Dimension for each dimension of the NCube. Dimension includes an attribute rank whose value corresponds to the dimension rank of the variable as described in the NCube and either the attribute value providing the variable value (intersect point) for this cell OR a reference to a Variable which will contain the value for the intersect point.

Example:

Variable: Age
1 = Under 18
2 = 18 to 64 years
3 = 65 years and older

Variable: Sex
1 = Male
2 = Female

Variable: PoliticalParty
001 = Democrat
002 = Democratic Farmer Labor
003 = Republican
004 = Independent
005 = Green Party
…
560 = Whig

LogicalProduct Description:
NCube1 Age by Sex
    Dimension rank=”1” VariableReference = “Age”
    Dimension rank=”2” VariableReference = “Sex”

NCube2 Age by Sex by Political Party Affiliation
    Dimension rank=”1” VariableReference = “Age”
    Dimension rank=”2” VariableReference = “Sex”
    Dimension rank=”3” VariableReference = “PoliticalParty”

For NCube 1 the DataItem for “65 years and older, Female” would be identified as follow:

```
<DataItem>
        <Dimension rank=”1” value=”3”/>
        <Dimension rank=”2” value=”2”/>
```

For NCube 2 there are only as many records for a particular NCube as needed to capture the Political Parties present for the geographic area in order to avoid a predominately empty NCube content. For example, Democratic Farmer Labor is found only in Minnesota. The DataItem for “18 to 64 years, Male, [Any PoliticalParty]” would be identified as follow:

```
<DataItem>
        <Dimension rank=”1” value=”2”/>
        <Dimension rank=”2” value=”1”/>
        <Dimension rank=”2”>
                <VariableReference isReference=”true”>
                        </ID>PolticalParty</ID>
                </VariableReference>
        </Dimension>
```

If the value of the DataItem referencing the Variable PolitcalParty is “005” then the count found in the DataItem described above will be the count of Males 18 to 64 years of age who identified as Green Party, if “002” those who identified with Democratic Farmer Labor. This structure is used primarily for legacy storage structures where space was major issue. It is also found in historical files covering topics that change dramatically in the content of the variable values over time.

Dimension provides the link between the DataItem being discussed and its position in the NCube matrix. A DataItem as a cell of an NCube can contain one or more measures as well as one or more attributes. The attributes described at this level apply to the specific DataItem only. An example of this is cell level suppression flags that are stored in the data file. It is identical in structure to that found at the NCubeInstance level. It provides a reference to the Attribute as described in the NCube logical product description, information on the physical storage location using PhysicalLocation or the value of the attribute. In general the point of having an attribute at this level is that it varies with each instance of the cell, but the structure allows for declaring a set value in the metadata at this point.

Measure is also repeatable to allow for multiple measures (count, percent, cumulative percent, etc.) to be attached to the DataItem. Each Measure contains a MeasureReference and PhysicalLocation. Note that MeasureReference extends ReferenceType by providing an attribute arrayOrder (integer with a default setting of “0” assuming an array base of ‘0”). Depending on the storage of the data, multiple measures for a single DataItem can be stored separately or as an array with a single PhysicalLocation. DDI allows for both storage structures.

If the measures have separate physical location information (different StartPosition etc.) the element Measure in DataItem is repeated and a single MearsureReference within Measure is provided. The arrayOrder on MeasureReference remains at “0”.

```
<DataItem>
    <Measure>
        <MeasureReference arrayOrder=”0”>
            </ID>COUNT</ID>
        </MeasureReference>
        <PhysicalLocation>
            …..
        </PhysicalLocation>
    </Measure>
    <Measure>
        <MeasureReference arrayOrder=”0”>
            </ID>PERCENT</ID>
        </MeasureReference>
        <PhysicalLocation>
            …..
        </PhysicalLocation>
    </Measure>
<DataItem>
```

If the measures are stored as an array at a single physical location information (single StartPosition etc.) the element Measure in DataItem is entered once using multiple MearsureReferences within Measure and a single PhysicalLocation. The arrayOrder on MeasureReference would be changed to reflect the position of the referenced measure in the array.

```
<DataItem>
    <Measure>
        <MeasureReference arrayOrder=”0”>
            </ID>COUNT</ID>
        </MeasureReference>
        <MeasureReference arrayOrder=”1”>
            </ID>PERCENT</ID>
        </MeasureReference>
        <PhysicalLocation>
            …..
        </PhysicalLocation>
    </Measure>
<DataItem>
```

This is the basic structure of NCube storage structures. Tabular and Inline descriptions will focus on the differences with this basic structure.

8.2.4 Tabular NCube Record Layout

A tabular layout is assumed to be a two dimensional layout on a spreadsheet or print storage that is defined by columns and rows. A spreadsheet can contain multiple tables on a single sheet and the table may be located anywhere on the sheet. In addition a “Table” may contain more than a single NCube, either hinged along a common dimension (see the example in section 7.5) or tightly interlaced in a specialized layout. The RecordLayout found in tabular_ncube_recordlayout.xsd can accommodate multiple NCubes found on a single spreadsheet layout. In addition to the list of NCubeInstances, the tabular description provides a TopLeftTableAnchor which gives the column and row of the upper left corner of the table being described. These are expressed as integers to provide a standard mappable structure for this content.

The NCubeInstance is identical to that of the basic NCube RecordLayout until one gets to the level of the PhysicalLocation. Tabular does not use the standard PhysicalLocation, but a specialized extension which adds the elements Column and RowSequence. Column is the column in which the DataItem will be located. RowSequence is the Row number within the repeating sequence which holds the content of the DataItem. For the table Urban/Rural by Age by Nativity by Sex (cell contents are DataItem coordinate values):

TABLE 1			Native Born		Foreign Born	
			Male	Female	Male	Female
Minnesota	Urban	Under 18	1,1,1,1	1,1,1,2	1,1,2,1	1,1,2,2
		18 to 64	1,2,1,1	1,2,1,2	1,2,2,1	1,2,2,2
		65 and over	1,3,1,1	1,3,1,2	1,3,2,1	1,3,2,2
	Rural	Under 18	2,1,1,1	2,1,1,2	2,1,2,1	2,1,2,2
		18 to 64	2,2,1,1	2,2,1,2	2,2,2,1	2,2,2,2
		65 and over	2,3,1,1	2,3,1,2	2,3,2,1	2,3,2,2

This shows a single record sequence so that the Column and RowSequence values for Rural, 18 to 64, Foreign Born, Male would be Column = 3 RowSequence = 5. Note that the TopLeftTableAnchor would have been stated as Column=3 and Row=3 (assuming the “TABLE 1” is located at Column=1 and Row=1 with an array base of 1). If this table is repeated for all states as opposed to including the states in the NCube structure, the location codes would not change with the repetitions of each new case.

TABLE 1			Native Born		Foreign Born	
			Male	Female	Male	Female
Minnesota	Urban	Under 18	1,1,1,1	1,1,1,2	1,1,2,1	1,1,2,2
		18 to 64	1,2,1,1	1,2,1,2	1,2,2,1	1,2,2,2
		65 and over	1,3,1,1	1,3,1,2	1,3,2,1	1,3,2,2

	Rural	Under 18	2,1,1,1	2,1,1,2	2,1,2,1	2,1,2,2
		18 to 64	2,2,1,1	2,2,1,2	2,2,2,1	2,2,2,2
		65 and over	2,3,1,1	2,3,1,2	2,3,2,1	2,3,2,2
Mississippi	Urban	Under 18	1,1,1,1	1,1,1,2	1,1,2,1	1,1,2,2
		18 to 64	1,2,1,1	1,2,1,2	1,2,2,1	1,2,2,2
		65 and over	1,3,1,1	1,3,1,2	1,3,2,1	1,3,2,2
	Rural	Under 18	2,1,1,1	2,1,1,2	2,1,2,1	2,1,2,2
		18 to 64	2,2,1,1	2,2,1,2	2,2,2,1	2,2,2,2
		65 and over	2,3,1,1	2,3,1,2	2,3,2,1	2,3,2,2

8.2.5 Inline NCube Record Layout

Again, this layout is essentially similar to the basic NCube Record Layout, but in this case rather than a location being specified for a DataItem’s attribute and/or measure, the value is provided inline in the instance. This results in the Dataitem gaining an optional attribute xs:lang to provide a language flag at the DataItem level. There is no PhysicalLocation provided. Attribute retains its optional Value element which becomes required, and Measure replaces PhysicalLocation with a required Value element. Note that Value is of type xs:string to allow for alphanumeric or symbol content. It becomes very important to be sure that all DataItems either use the default data type and other structural defaults, or declares them at the attribute level. Inline NCube RecordLayout is the functional equivalent of DataSet for aggregate data.

8.2.6 Proprietary Record Layout (BETA)

This provides a generic structure for describing record layouts for proprietary software, in particular statistical analysis software. In addition to the elements inherited from BaseRecordLayout, ProprietaryRecordLayout includes the following elements.

Element	**Use**
Software	Standard software identification structure for the software used by this data structure
DataItemAddress	Description of how data items are addressed within the file, for example by Variable ID or by Variable Name
DefaultNumericDataType	Declares the most common data type used for numeric data using a controlled vocabulary
DefaultTextDataType	Declares the most common data type used for text data using a controlled vocabulary
DefaultDateTimeDataType	Declares the most common datetime type used for text data using a controlled vocabulary
CHOICE:	

CodeDataAsNumeric	Use indicates that variables using CodeRepresentation should normally be treated as numeric data and declares the most common data type using a controlled vocabulary
CodeDataAsText	Use indicates that variables using CodeRepresentation should normally be treated as text data and declares the most common data type using a controlled vocabulary
ENDCHOICE	
ProprietaryInfo	This is a name value pair providing information proprietary to the software package.
DataItem	Contains a VariableReference a ProprietaryDataType to override the default, a ProprietaryOutputFormat to designate the desired display or other output format, and ProprietaryInfo at the DataItem level

7.7 Physical Instance

Physical Instance has a one-to-one relationship with an actual physical data file. The single exception is duplicate copies of the same file. A PhiscalStructure may have zero (as is the case when using dataset), one, or hundreds of data files associated with it. Common reasons for this are cases where each record type or record segment is stored in a separate file or when large datasets result in subsets of records by spatial, temporal, or topical divisions. For example, the 2000 US Census Summary File 4 has a separate data file for each state by each record segment by one or a range of characteristic iteration values. This results in a collection of over one million separate files in order to cover the US. While this is an extreme case, it is also not uncommon for an archive to acquire a single dataset such as the Eurobarometer and immediately sort it into a file per country to facilitate use by their researchers.

Physical Instance is designed to capture these types of divisions as well as contain the summary and category statistics related to the full dataset and/or those specific to the particular subset of records.

7.7.1 Top Level Elements

The Physical Instance has the standard common reusable elements and is the module that most frequently makes use of Coverage to impose coverage constraints. The coverage of the Physical Instance will often be a subset of the study coverage. In the past, coverage constraints of a data file were often only noted by cryptographic file names. Physical Instance allows for clear evidence of the subset included in the related data file. Additional elements identify the Physical Data Product that describes the record(s) contained in the file as well as the name of the data file itself, its location, and any additional copies. Attributes let you indicate which is a master file (as opposed to a backup or other copy), and note the URI if it is publicly available. In addition to the URI of the data file, a location and path can be provided as a file may not be available directly due to access restrictions. (It may, in fact be located on a DVD sitting in a safe, etc.).

In addition to these standard forms of identification, DDI has provided a generic structure for capturing the “fingerprint” of the data file. This includes a Value for the fingerprint, the AlgorithmSpecification, and the AlgorithmVersion. The Fingerprint is repeatable to allow for use of multiple algorithm specifications.

7.7.2 Gross File Structure

This section contains information unique to the individual data file and how it was created.

ELEMENT	USEAGE
PlaceOfProduction	Where the file was produced.
ProcessingCheck	Description of any processing checks that were done when the file was made [or subsequent checks].
ProcessingStatus	Many files go through stages in production and the current stage should be noted here. Earlier stages should have been noted in the set of ProcessingCheck entries.
CreationSoftware	The name of the software used to create the file including the name, version, description, and date of the software.
CaseQuantity	Number of cases in the file.
OverallRecordCount	Total number of records in the file (a case may have more than one record).

7.7.3 Statistics

Summary and category statistics for a data file are entered in the physical instance as the values change when full datasets are subset. Statistics may be entered inline in the metadata or may exist as a separately described dataset (common for large complex datasets expressing summary or category statistics for filtered variables such as each variable by country). The StatisticalDataFile reference may be to a physical instance that contains the statistics inline or that represents the data file containing the statistics (noted by an attribute).

Statistics captured inline are organized by variable. The structure provides for total responses, weight references, handling of missing category, additional summary statistics, and category statistics. Category statistics can be weighted, and can be presented as multiple forms of statistics (count, frequency, cumulative frequency, etc.), and filters. A filter allows you to designate a single layer cross-tabulation. For example:

Variable: Sex				COUNT
	Category:			
		Male		COUNT
		Female		COUNT
	Filter Variable: Country			
		Germany		
			Male	COUNT
			Female	COUNT
		France		
			Male	COUNT
			Female	COUNT

8.0 Group, Resource Package, and Comparison

The schema group.xsd encompasses a number of the features of DDI 3.0 that allow it to capture the life cycle of data. Group provides an umbrella structure to pull together two or more studies into a structured series or unstructured group. It provides basic information on relationships among members of the group that affect processing decisions based on a required attribute grid (see DDI 3.0 Technical Specification Part I: Overview, Appendix Two) as well as detailed information on comparable relationships between and among the studies in the group. In addition, Group contains a specialized structure called ResourcePackage that allows for publishing maintainable objects (schemes and schemas) outside of a StudyUnit or Group.

Studies can be grouped for a number of reasons but generally fall into the category of grouping by design or ad hoc groups. Grouping by design takes place when studies are either intended to be a series or when a repetition of the study takes place. The key factor is that the second study in the series is intended to inherit features of the first study (questions, variables, study design, universe, etc.) for the purpose of comparability. Group allows you to define which parts of the major components are shared, where overrides take place, and how to relate or link data in one study to data in a subsequent survey.

When using inheritance within groups to show comparability – or even just to re-use metadata – it is important to understand how local overrides work, as this can impact the way the metadata is grouped. Within each group, all metadata is inherited down the grouping structure. At any level, it is possible to override any inherited metadata using the Add, Replace, or Delete attributes which are found on the IdentifiableType, VersionableType, and MaintainableType structures. To override an inherited structure, it should have the appropriate ID structure given for it, and then have the Replace element specified. To delete inherited metadata, use a similar technique but employ the Delete element. Once replaced or deleted, it is the modified form of the metadata which is inherited down the grouping structure.

Note that when referencing metadata that is subject to local overrides, it may be necessary to specify the exact module being referenced – otherwise, local deletions and overrides won't be referenced.

Ad hoc groups are collections of studies that have been grouped to meet specific needs of the archive, data service, or user. These groups do not support comparison through inheritance as they were developed as separate studies and grouped later to meet specific needs. Without the use of inheritance, comparability must be described explicitly using the schema Comparative. Currently, comparison is enabled for the following complex elements: Universe, Concept, Question, Category, CodingScheme, and Variable. All comparisons are pairwise, and with the exception of CodingScheme each comparison notes a source and target item, the relationship (map) type, and a definition of any differences when the map type is less than a full equality. CodingScheme provides options for describing code relationships between two coding schemes. Options include a human-readable description of the translation process (Source code 1 through 3 equal Target code 1), a command line for a specified command language, or the use of the GenerationInstruction from DataCollection. For example a direct mapping of each source code value to its target value could be declared.

SOURCE		**TARGET**
Code 1 Never Married	recode to	Code 1 Single
Code 2 Divorced	recode to	Code 1 Single
Code 3 Widowed	recode to	Code 1 Single
Code 4 Married	recode to	Code 2 Married

Alternatiively, GenerationInstruction could be used to identify the Source Variable and provide the command code If Source >= 1 and <= 3 Target = 1; If Source = 4 Target = 2.

Correspondence does not limit itself to equivalency but captures a text description of Commonality and Difference as well as a CommonalityTypeCoded, CommonalityWeight, and a UserDefinedCorrespondenceProperty. The following table provides some guidance in classifying the level of comparison for Universe, Concept, Question and Variable as well as the associated Category and Code Schemes.

Structure of Comparisons

Similar: Denotes a close but not exact relationship; requires description of difference.
Different: Denotes non-equivalent coding scheme; requires coding instructions to create equivalency.

Questions and Variables are assumed to use the categories and coding scheme of the source item.

Comparison Map	**Textual Content of main body**		**Category**		**Code Scheme**		***ACTION***
	Same	*Similar*	*Same*	*Similar*	*Same*	*Different*	
Universe	X		----	----	----	----	Enter and Flag as Identical
		X	----	----	----	----	Flag as similar and note differences in human-readable and optional repeatable machine-actionable string.
Concept	X		----	----	----	----	Enter and Flag as Identical.
		X	----	----	----	----	Flag as similar and note differences in human-readable and optional repeatable machine-actionable string.
Question	X		X		X		Enter and Flag as Identical; include concepts and coding schemes used.
	X		X			X	Source and target relationship for question and category scheme must be the same. Reference harmonized coding scheme.
	X			X	X		Flag as similar and note category differences in human-readable and optional repeatable machine-actionable string. Assume categories related to codes are those of the source code value.
	X			X		X	Flag as similar and note category differences in human-readable and optional repeatable machine-actionable string. Source and target relationship for question and code scheme must be the same. Reference harmonized coding scheme.

		X	X		X		Flag as similar and note text differences in human-readable and optional repeatable machine-actionable string. Assume categories related to codes are those of the source code value.
		X	X			X	Flag as similar and note text differences in human-readable and optional repeatable machine-actionable string. Source and target relationship for question and code scheme must be the same. Reference harmonized coding scheme.
		X		X	X		Flag as similar and note text and category differences in human-readable and optional repeatable machine-actionable string. Assume categories related to codes are those of the source code value.
		X		X		X	Flag as similar and note text and category differences in human-readable and optional repeatable machine-actionable string. Source and target relationship for question and code scheme must be the same. Reference harmonized coding scheme.
Variable	X		X		X		Enter and Flag as Identical; include concepts and coding schemes used.
	X		X			X	Source and target relationship for variable and code scheme must be the same. Reference harmonized coding scheme.
	X			X	X		Flag as similar and note category differences in human-readable and optional repeatable machine-actionable string. Assume categories related to codes are those of the source code value.
	X			X		X	Flag as similar and note category differences in human-readable and optional repeatable machine-actionable string. Source and target relationship for variable and code scheme must be the same. Reference harmonized coding scheme.
		X	X		X		Flag as similar and note text differences in human-readable and optional repeatable machine-actionable string. Assume categories related to codes are those of the source code value.

		X	X			X	Flag as similar and note text differences in human-readable and optional repeatable machine-actionable string. Source and target relationship for variable and code scheme must be the same. Reference harmonized coding scheme.
		X		X	X		Flag as similar and note text and category differences in human-readable and optional repeatable machine-actionable string. Assume categories related to codes are those of the source code value.
		X		X		X	Flag as similar and note text and category differences in human-readable and optional repeatable machine-actionable string. Source and target relationship for variable and code scheme must be the same. Reference harmonized coding scheme.

9.0 Step-by-Step Sequence to Create a DDI File for a Simple Instance

These sections should be completed in the following order to ensure that fields containing information that is required (linked) from other elements is available when needed. Note that the list is ordered so that none of the entries requires a section lower down in the list for its creation.

Sequence No.	**Section**	**Content**	**Elements/sections referencing items**
1	**UniverseScheme** All universe items in a structured hierarchy	Universe identification is done by referencing a single universe structure. This may be built up as you go along, but the top level universe should be entered as a container for sub-universes and as the content of the study unit universe.	UniverseReference in studyunit, ControlConstruct, Variable, NCube, UniverseMap. Preexistence of this element also helps in defining coverage and analysis units

2	**ConceptScheme** A structured list of all concepts of the study	Ideally this should be as detailed as practicable to support ISO 11179. At a minimum there must be at least a single comprehensive concept as ConceptReference is a required element in Question and Variable.	ConceptReference in Question, Variable, and ConceptMap
3	**Organization** and **Individual** A listing of all organizations and individuals involved in the life cycle of the study to date	The organization module contains information on organizations, individuals and their structural roles and relationships. This information is housed separately and referenced by a number of element types.	Citation may reference producers, publishers, distributors, etc. Funding information found in studyunit, datacollection, archive, and group reference organizations and individuals. At minimum you will need a listing of the organization or individual who acts as the maintenance agency and for the archive (may be the same). This can be held in a public registry of DDI organizations.
4	StudyUnit **Citation / Abstract / Purpose** This is a required part of the study unit and represents the fact that a study unit must exist in order to create the specified section	A basic studyunit is the broad description of a simple study. Intellectually, the collection of data and creation of a data file are done within the construct of a study.	The following module cannot exist without a studyunit. Archive must have a study unit to attach to as it describes an archive holding of some study. Group, by definition is two or more study units.
5	StudyUnit **Coverage** A definition of the topical, temporal, and geographic coverage of the study	Coverage at this level is the top level container describing aspects of coverage. All other instances of coverage are either subsets of this	DataCollection, LogicalProduct, PhysicalDataProduct, PhysicalInstance, Group, and the majority of sub-schemas or sub-

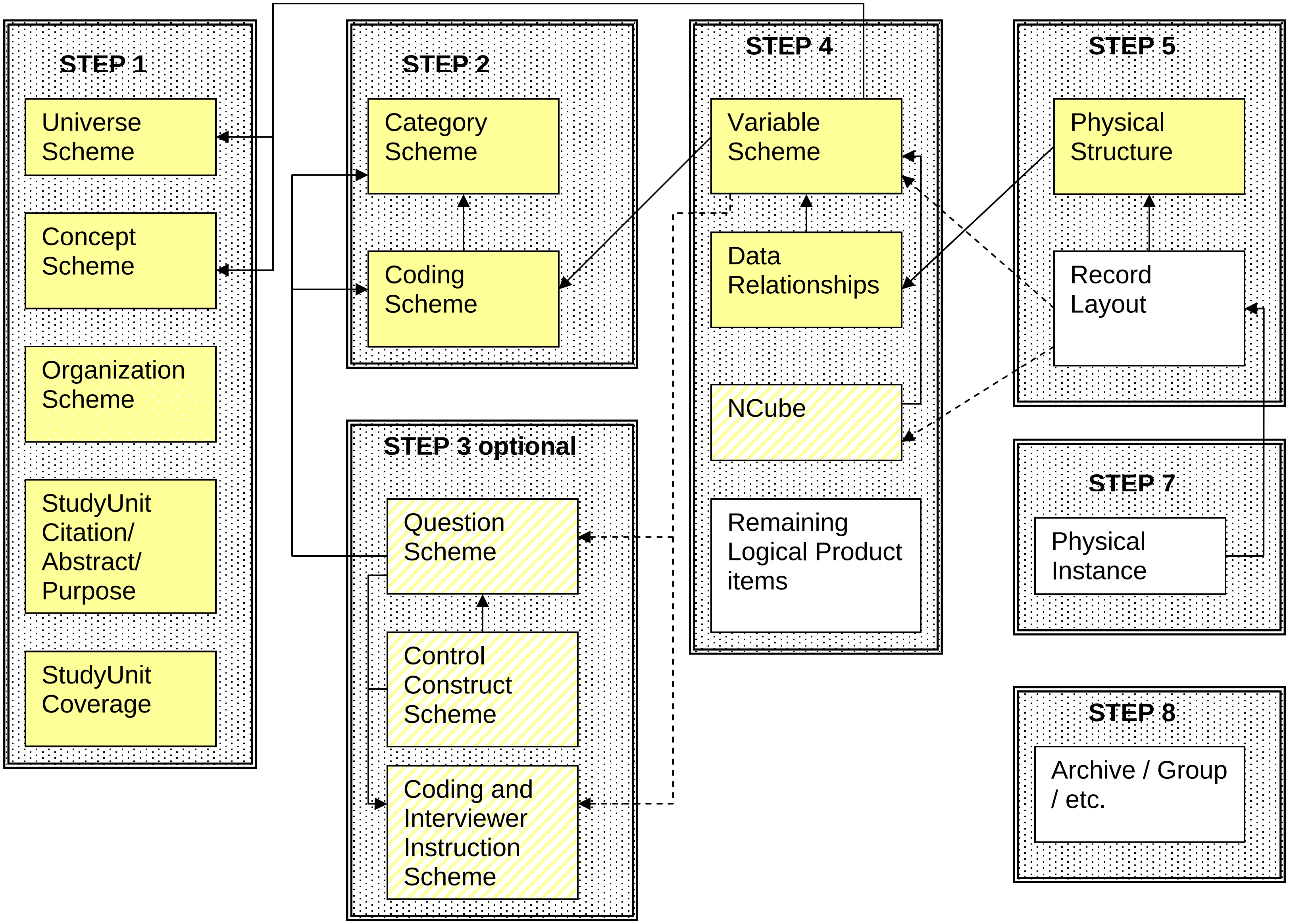
		definition, or in the case of Group, the inclusive combination of the coverage of the study units in the group. Note that if GeographicStructure and GeograhicLocation are used, the Schemes are located in ConceptualComponnents. They may also be published as an external resource.	modules.
6	LogicalProduct **CategoryScheme** This contains all categories and their definitions used in the study	Categories are defined once and referenced either directly or through their representational code.	Used to construct Question ResponseDomains, CodeSchemes, and Variables by direct reference or via CodeSchemes.
7	LogicalProduct **CodeScheme** Connects codes to categories	Organizes categories into structured or unstructured groups and provides the representational code (for example “0 = Male”). For structured CodeSchemes, information on subgroups and relationships is provided.	Question ResponseDomain and Variable Representation
8	DataCollection **QuestionScheme** Contains question text and response domain information	If Instrument will be included in your instance or if Variable will need to reference the question, the QuestionScheme must be completed.	ControlConstruct, CodingInstructions (ProcessingEvent) (possibly), and Variable (optional)
9	DataCollection ProcessingEvent **Coding** Explains the process of altering the question response to obtain	This includes general coding instructions, recodes, inclusion of administrative or information from outside of the questionnaire, or	Question (interviewer instructions), ControlConstruct (either), Variable (coding instruction)

	the content of the variable. **InterviewerInstructionScheme** contains all the interviewing instructions including additional descriptive information and visible routing instructions used when completing the questionnaire.	other derived or generated content for the variable. All descriptions are housed in this location and referred to by elements using the process described. All interviewer instructions are held in the scheme and included by reference by the question or the control construct.	
10	**VariableScheme** Defines the intellectual content and structure of microdata elements and dimensions for aggregate data matrixes (NCubes)	The variable scheme defines individual variables. They are used to construct NCubes and provide a single source of intellectual content regardless of the storage structure. Variables can be grouped in several ways including those contained within a single type of record.	NCube, VariableGroup, DataRelationships, DataItem, SummaryStatistic, CategoryStatistic
11	LogicalProduct **DataRelationships** Defines the identification of each record type, information needed to identify a specific case, and links between record types	The structural information provided here is a map to assist in record identification, selection, and linking. This is the intellectual map of the types of structures and linkages supported by the variable content of the record. Actual physical layout is described separately, but use the information provided here, simply adding the specifics of how the physical file dealt with the intellectual structure.	PhysicalStructure. Required link to the LogicalRecord
12	**NCubeScheme** Describes aggregate data	The NCube is defined by its universe and dimensions. Each	DataItem

	tables/matrices	cell of the NCube is identified by its matrix coordinate pattern.	
13	**PhysicalStructure/ PhysicalRecordSegment** Identifies each physical storage record type and how it uses the intellectual links between record types as described in DataRelationships	A data set is made up one or more physical record types stored in one or more physical data file structures.	RecordLayout contains a required link between the RecordLayout and the PhysicalRecordSegment.
14	**RecordLayout** desctibes the physical layout of data items within a record OR provides the data inline	The DataItems describe the physical location of the variable or NCube cell being described on a specified physical record type. The PhysicalInstance may contain one or multiple physical record types and must identify which of these types described in PhysicalData that it contains.	PhysicalInstance contains a required link to one or more RecordLayouts.

The following diagram identifies sections of DDI in the order they need to be completed. The gray boxes are major steps and include one or more sections of DDI elements. The solid yellow boxes represent element sets that will be required for references or other use later in creating the full instance. The boxes with yellow diagonal bars are required if you provide references (such as a reference from a variable to a question) or have this feature (NCubes). Arrows indicate references from one element set to another. Dotted line arrows indicate that the inclusion of this information is by choice. If you do not have the element set you do not need the references. References from STEP 8 are dependent upon what features you choose to include; however, at this point the required material is available for reference.

Data Documentation Initiative



STEP 1

Universe Scheme

Concept Scheme

Organization Scheme

StudyUnit Citation/ Abstract/ Purpose

StudyUnit Coverage

STEP 2

Category Scheme

Coding Scheme

STEP 3 optional

Question Scheme

Control Construct Scheme

Coding and Interviewer Instruction Scheme

STEP 4

Variable Scheme

Data Relationships

NCube

Remaining Logical Product items

STEP 5

Physical Structure

Record Layout

STEP 7

Physical Instance

STEP 8

Archive / Group / etc.