Structure content of database for documentation in addition to below

- Valid DDI Version Range: Begin End
- Date
- Version number of metadata content
- ID
- Author 1..n
- CONTENT

Template for Parts I and II

- Cover
- Acknowledgements
- License
- Table of contents
- Sections (multiple)
- Appendices
- Index

Part I: Overview

Background

Relationship to other standards-more relation information detail, mapping where practical, transition points

- DDI-C
- ISO/IEC 11179 – elements in ISO/IEC 11179 and DDI – use of ISO/IEC 1179-5 profile
- ISO 19115
- SDMX

Move details of common xml structures (xml:lang, datetime etc) into User Guide. Just provide W3C etc link in Overview

List of codevalue types

Technical structures

Modules/Schemes – structure

- Include top level objects as structure

Special element types used in multiple locations:

- Identifiable, Versionable, Maintainable, Reference, Scheme Reference
- Date types
- String types
    - XHTML
- Dublin Core
- Notes
- Other Materials
- Citation
- Name, Label, Description
- Scheme structures

Alpha listing of major complex elements

    - Parent maintainable
    - Namespace
    - Name
    - Nested framework
    - Required Referenced objects
    - Optionally Referenced object
    - Purpose
    - Content documentation
    - Examples (snippets)

Change list (appendix):

- Element and path
- Backward compatible?
- Bug reference
- Update process

Index of all elements linking back to description pages

Designate new elements by font?

- Produce from structured XML database and transform each new edition. For new elements use a date designator to identify “new” elements / changes

Part II: User Guide

Focus on use case approach for various cases at different points in the life cycle

Provide examples for non-standard situations

Reference sections of Part I rather than repeat

- Simple study – legacy
- Simple study from investigator prospective
- Creating a questionnaire
- Managed conceptual components (universe, concept, geographic structure/location, data element)
- Archival management (Archive and Local Holding Package)
- Do not replicate purpose of Best Practice or Use Case papers but reference location as an additional resource
- More step by step instructions including:
    - Case description
    - Type of use supported by content
    - Reference to related sections of Part I
    - What to enter
    - Why it is needed / How it's used
    - Special case examples
    - DDIProfile

OVERVIEW EXAMPLE of a complex element display

Alpha listing of complex elements

- Parent maintainable
- Namespace
- Name
- Nested framework
- Required Referenced objects
- Optionally Referenced object
- Purpose
- Content documentation
- Examples (snippets)

Complex Element

DataRelationship	
Namespace: **l** *Parent Maintainable:* **LogicalProduct**	**Versionable**
Required Referenced Objects:	*Optionally Referenced Objects:* LogicalRecord NCube NCubeScheme Variable VariableScheme

l:DataRelationship
- l:DataRelatiohshipName (0..n)
- r:Label (0..n)
- r:Description (0..n)
- l:LogicalRecord (1..n)
  - @hasLocator (default=”false”)
  - @variableQuantity
  - l:LogicalRecordName (0..n)
  - r:Label (0..n)
  - r:Description (0..n)
  - l:VariableValueReference (0..1)
    - l:VariableReference (1..1)
    - l:Value (0..n)
  - l:SupportForMultipleSegments (0..1)
    - l:VariableReference (1..1)
    - l:Value (0..n)
  - l:CaseIdentification (0..n)
    - @isPrimary (default=”true”)
    - l:VariableReference (1..1)
    - l:VariableSpecification (1..1)
      - CHOICE (1..1)
        - l:SelectorVariableReference

```
            l:CaseSpecification
               l:Value (1..1)
               l:VariableReference
   CHOICE
      l:VariablesInRecord
         @allVariablesInLogicalProduct (default="true")
         l:VariableSchemeReference (0..n)
         l:VaraibleUsedReference (0..n)
      l:NCubesInRecord
         @allNCubesInLogicalProduct  (req)
         l:VariablesInRecord (0..1)
         l:NCubeSchemeReference (0..n)
         l:NCubeReference (0..n)
   ENDCHOICE
 l:RecordRelationship (0..n)
    @type (default="Equal")
    l:RecordRelationshipName (0..1)
    r:Label (0..n)
    r:Description (0..n)
    l:SourceRecord (1..1)
       @relation (required)
       l:LogicalRecordReference (1..1)
       l:LinkVariableReference (1..1)
    l:TargetRecord (1..1)
       l:LogicalRecordReference (1..1)
       l:LinkVariableReference (1..1)
```

Data Relationship

DataRelationship defines the contents of a logical record in terms of the variables or NCubes it contains, describes how to identify a unique case within a logical record type, and how to relate two or more logical records. This section is optional only because a logical product may not contain variables or NCubes, simply a category and/or code scheme used by a set of questions. A link to a LogicalRecord in a DataRelationship is required by all PhysicalStructure descriptions. In its simplest form a DataRelationship for a microdata file (variables) must contain the following:

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

This states that all the variables in the logical product are part of a single logical record which has no variable field that identifies its record type. This is the structure used by most simple surveys. However, DataRelationship can also provide the detailed information needed to describe the content and

relationship of a complex set of logical records whose contents may be described in one or more logical products. The two things that it does not do is to describe the storage order of those variables or differentiate between a single logical record stored as a single string and one stored as a series of segments. Both of these aspects are described in the PhysicalDataStructure. DataRelationship deals only with the intellectual content of a logical record and relationships between logical records.

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

Logical Record

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

The LogicalRecord must provide either VariablesInRecord for microdata files or NCubesInRecord for aggregate files. NCubesInRecord allow for identifying both NCubes and Variables to accommodate those files where case identification is provided in a variable string that is not described as part of the NCube structure. It is not used to list the variables that are used as dimensions or measures unless there is data in a file associated specifically with the variable. Both Variables and NCubes can be identified by a full scheme or schemes (allows for exclusions) and by individual variable references. If all the variables or NCubes with the logical product housing the DataRelationship are used in the logical record the Boolean attribute allVariablesInLogicalProduct or all NCubesInLogicalProduct can simply be set to “true” and no further definition is required.

Record Relationship

As with all other relationship definitions RecordRelationship is pair-wise standing a Source and Target, each stating their variable location, value if appropriate and a relationship type (parent, child, or sibling). Note that this is a single variable reference for each Source and Target. If the link key is a concatenation of two or more variables, you must create a concatenated variable to use for this reference. Once the Source and Target have been identified the relationship between the Source and Target variable values

can be set to Equal (default), GreaterThan, LessThan, GreaterThanOrEqual, LessThanOrEqual, or NotEqual. This simple structure and pair-wise approach provides consistent linking information for the simplest to the most complex files.

EXAMPLE

This example shows a common multi-record data product with a single variable link between records. Special use cases of this complex element, in particular complex case identification and complex record relationships are found in Part II: User's Guide.

```
<l:DataRelationship isVersionable="true" id="DR_1" version="1.0">
  <l:DataRelatiohshipName xml:lang="en">US PUMS</l:DataRelationshipName>
  <r:Label xml:lang="en">United States Census Public Use Micro Sample</r:Label>
  <r:Description xml:lang="en">The US Census Bureau Public Use Microdata Sample from the
  Decennial Census. The PUMS is a hierarchical structure with one household record per housing
  unit and individual person records for each person in the household.</r:Description>
  <l:LogicalRecord isIdentifiable="true" id="LR_H" hasLocator="false" variableQuantity="53">
    <l:LogicalRecordName xml:lang="en">HHREC</l:LogicalRecordName>
    <r:Label xml:lang="en">Household Record</r:Label>
    <r:Description xml:lang="en">Household record containing all variables specifi to the
    household and common to the members of the household.</r:Description>
    <l:VariableValueReference>
      <l:VariableReference isReference="true">
        <r:URN>urn:ddi:us.uscb:VariableScheme:VS_1:2.0:Variable:V1:1.0</r:URN>
      </l:VariableReference>
      <l:Value>H</l:Value>
    </l:VariableValueReference>
    <l:CaseIdentification isPrimary="true">
      <l:VariableReference isReference="true">
        <r:URN>urn:ddi:us.uscb:VariableScheme:VS_1:2.0:Variable:V2:1.0</r:URN>
      <l:VariableSpecification>
      <l:SelectorVariableReference>
        <l:VariablesInRecord allVariablesInLogicalProduct="false">
          <l:VariableSchemeReference isReference="true">
            <r:URN>urn:ddi:us.uscb:VariableScheme:VS_1:2.0</r:URN>
          </l:VariableSchemeReference>
        </l:VariablesInRecord>
      </l:SelectorVariableReference>
      </l:VariableSpecification>
    <l:CaseIdentification isPrimary="true">
  </l:LogicalRecord>
  <l:LogicalRecord isIdentifiable="true" id="LR_P" hasLocator="false" variableQuantity="53">
```

```
  <l:LogicalRecordName xml:lang="en">PREC</l:LogicalRecordName>
  <r:Label xml:lang="en">Person Record</r:Label>
  <r:Description xml:lang="en">Person record containing all variables specific to the
  person.</r:Description>
  <l:VariableValueReference>
    <l:VariableReference isReference="true">
      <r:URN>urn:ddi:us.uscb:VariableScheme:VS_1:2.0:Variable:V1:1.0</r:URN>
    </l:VariableReference>
    <l:Value>P</l:Value>
  </l:VariableValueReference>
  <l:CaseIdentification isPrimary="true">
    <l:VariableReference isReference="true">
      <r:URN>urn:ddi:us.uscb:VariableScheme:VS_1:2.0:Variable:V2:1.0</r:URN>
    <l:VariableSpecification>
    <l:SelectorVariableReference>
      <l:VariablesInRecord allVariablesInLogicalProduct="false">
        <l:VariableSchemeReference isReference="true">
          <r:URN>urn:ddi:us.uscb:VariableScheme:VS_1:2.0</r:URN>
        </l:VariableSchemeReference>
      </l:VariablesInRecord>
    </l:SelectorVariableReference>
    </l:VariableSpecification>
  <l:CaseIdentification isPrimary="true">
</l:LogicalRecord>
<l:RecordRelationship type="Equal">
  <l:RecordRelationshipName xml:lang="en">H P link</l:RecordRelationshipName>
  <r:Label xml:lang="en">Link between household and person record</r:Label>
  <r:Description xml:lang="en">The single variable link between the parent household record and
  all person records within the household. A one-to-many relationship</r:Description>
  <l:SourceRecord relation="Parent">
    <l:LogicalRecordReference isReference="true">
      <r:URN>urn:ddi:us.uscb:LogicalProduct:LP_1:2.0:LogicalRecord:LR_H:1.0</r:URN>
    </l:LogicalRecordReference>
    <l:LinkVariableReference isReference="true">
      <r:URN>urn:ddi:us.uscb:VariableScheme:VS_1:2.0:Variable:V3:1.0</r:URN>
    </l:LinkVariableReference>
  </l:SourceRecord>
```

```
    <l:TargetRecord>
      <l:LogicalRecordReference isReference="true">
        <r:URN>urn:ddi:us.uscb:LogicalProduct:LP_1:2.0:LogicalRecord:LR_P:1.0</r:URN>
      </l:LogicalRecordReference>
      <l:LinkVariableReference isReference="true">
        <r:URN>urn:ddi:us.uscb:VariableScheme:VS_1:2.0:Variable:V3:1.0</r:URN>
      </l:LinkVariableReference>
    </l:TargetRecord>
  </l:RecordRelationship>
</l:DataRelationship>
```