Data Documentation Initiative (DDI) Technical Specification

Part I:

Technical Documentation

Version 3.2

1

February 2014

Technical Document for DDI Version 3.2

Version 3:2
Date: February 15, 2014
Wendy Thomas, Arofan Gregory, J Gager, Jon Johnson, Joachim Wackerow

Contents

Overall documentation

Documentation of the DDI specification is provided in three ways:

Field Level Documentation

AUDIENCE: Developers, database developers, mappings, base level for content providers (what an object is in relation to parent and child elements)

This documentation is found within the DDI Schemas and displayed in the HTML documentation. It provides a brief description of the purpose and content of the object. Documentation found in the complex Type description will provide more detail than the element documentation. Within a complex type, the additional documentation of sub-elements will focus on its purpose within the context of the complex type.

Part I - Technical Documentation

AUDIENCE: Developers, integrated usage and applications for content providers

Organized by related sets of objects, e.g. Question Item, Question Grid, and Question Block, this documentation provides details of the structure and its intended application. Each set contains examples of usage. It contains information on the relationship of DDI to other standards, common XML structures used by DDI, design and consistency rules, description of major structural types (modules and schemes), technical features for identification and reference, basic types for dates and strings, and all major complex elements. The complex element content is organized alphabetically by set and an index is provided for all elements. This documentation also contains lists of: 3.1 to 3.2 changes, all unique element and attribute names, and elements by extension base (Identifiable, Versionable, Maintainable, Reference, CodeValue, etc.).

Part II - User Guide

AUDIENCE: Content providers, those focusing on specific applied uses of DDI

Provides instructions for navigating the HTML Field Level Documentation and reviews basic structural features focusing on their usage, such as exchange structures, organizing publication package content, managing data over time, common structure like strings, controlled vocabularies, dates, citation and coverage, notes and other material. This general section is followed by a set of user stories (applying DDI). The focus is on how the parts of DDI work together to describe the metadata and data for particular functions such as documenting a longitudinal study or developing a questionnaire. Wherever appropriate, Part I will reference the more detailed technical documentation in Part I.

1.0 - Coverage of DDI [for additional background information see Part II]

1.1 - Purpose of Part I Technical document

The intent of the Part I: Technical Document is to provide implementers with additional information on major complex elements and their components. For each complex object set this includes its namespace, parent maintainable, extension base, required referenced objects, optionally referenced objects, a brief layout of content and object cardinalities, detailed description, and example. An appendix is provided covering changes between 3.1 and 3.2 including backward compatibility and update processing as well as an index of all complex elements with a reference to the section and page number providing coverage of the element.

Users interested in the application of DDI to various Use Cases should see Part II: User Guide. Part II will provide references to technical features described in Part I.

1.2 - Organization of Part I

Part I: Technical Documentation is organized into four major areas, general description of DDI and the Part I technical document, relationship of DDI to other standards, general structures within DDI, and specific structures (complex element sets). Internal reference will be found in-line using the section number and title of the section placed within square brackets, i.e. [3.1 Identification and Reference Structures].

2.0 - Relationship to Other Standards

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

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

2.1 - DDI Codebook (DDI-C)

After conceptualizing the lifecycle model and the design rules for reuse, DDI-C content was distributed to the schemas comprising DDI-L. Specific mapping of objects from DDI-C to DDI-L brought to light a

number of specific issues which were then addressed during DDI –L revisions. While specific objects may not always have a specific 1:1 correlation in 3.0, the content of all 2.1 objects has been captured, often in greater detail or a more consistent manner than in earlier DDI versions.

During the development of DDI 3.2 changes were made in DDI-C to incorporate more content from the Generic Statistical Business Process Model (GSBPM) and to add objects that would ease the translation of DDI-C into DD-L. These objects were also added to DDI-L if not already present.

2.2 - Dublin Core and MARC

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

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

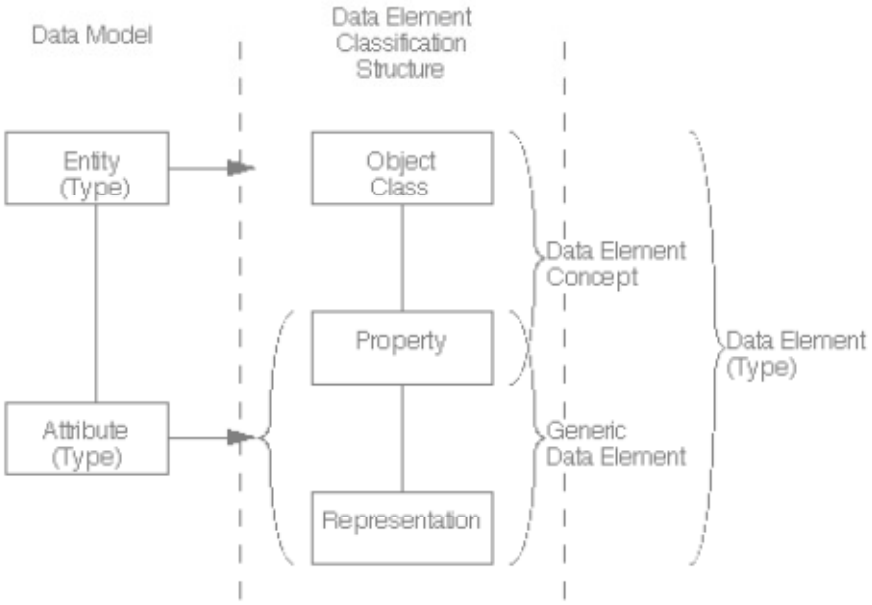
2.3 - GSIM (Generic Statistical Information Model)

GSIM provides a standardized set of information objects that flow through the process model in the creation of official statistics as represented by the Generic Statistical Business Process Model (GSBPM). GSBPM has been of interest to DDI since in conception. The DDI Generic Longitudinal Business Process Model (GLBPM) is modeled on the GSBPM which itself was informed by the DDI Lifecycle Model. DDI has worked with the UNECE in the development of GSIM to ensure that these two standards are compatible. In general, GSIM is a conceptual model whereas DDI is an implementation model. There is a shared interest in ensuring that DDI-L can be used as a means of implementing the GSIM conceptual model. Mapping work and the development of DDI profiles to support GSIM are taking place within that community.

A primary area for close mapping to the GSIM structure lies in that areas that relate to their reflection of the ISO/IEC 11179 Data Element Classification Structure. The terminology for the objects as well as the documentation for the objects reflects the GSIM model.

This standard describes the structure and content of a data element as the basic building block of information. DDI -L is particularly concerned with providing the information needed to populate an

ISO/IEC 11179 data element and support a registry structure. The following diagram provides the Data Element Structure.



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

In DDI terminology, the Object Class is defined by the universe, its Property is the concept, and the Representation is the Representation content used by the Variable that measures it. ConceptualComponent contains Universe and Concept definitions while Representation is described within the Variable. In most DDI instances it is the Variable that ties the three sections of this definition together. Note that if the Variable does not include a concept reference the instance is not compliant with ISO/IEC 11179.

DDI 3.2 has added the ability to create the more abstract (or reusable) descriptions of an ISO/IEC 11179 Data Element and Data Element Concept. These have been structured to align with the GSIM description (see GSIM in this section). The 3.1 element DataElementConcept has been replaced by a ConceptualVariable located within ConceptualComponent. This object links a Concept with a Universe creating the equivalent of the ISO/IEC 11179 Data Element Concept. A RepresentedVariable located in Logical Product, adds the expression of the representation to the concept and universe. This is the

reusable set of information for a Variable which is the implementation of a RepresentedVariable within a specific usage. The underlying RepresentedVariable may be referenced from the Variable.

ISO/IEC 11179

DDI-L reflects ISO/IEC 11179 structure in several respects. First, it maps the core structures Concept, Universe, ConceptualVariable, and RepresentedVariable to the ISO/IEC 11179 Data Element Classification Structure. The structure presented in DDI 3.2 aligns with the representation of this ISO standard through the GSIM structure. (See GSIM discussion within this section).

Secondly, DDI-L has adopted the combination of the Agency, ID, and Version as the full unique identification of an object. This is reflected in the identification model and content of the DDI URN.

Finally, DDI-L has incorporated the ISO/IEC 11179-5 structure of providing a Name, Label, and Description for all primary objects (maintainable and versionable) that are not considered publication structures. Publication structures have complete citation information plus an abstract. The objects with this common descriptive set are those that are commonly managed with registries over time.

ISO 19118 - Geography

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

In SpatialCoverage:

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

In GeographicResponseDomain:

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

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

SDMX

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

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

METS and PREMIS

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

PREMIS is a common implementation of Open Archive Information System (OAIS). There is a preliminary mapping of DDI-L to PREMIS objects. The focus of PREMIS is preservation and there are several elements where DDI-L does not provide controlled content. However, with the ability to publish controlled vocabularies external to the DDI specification, we should be able to address all but a few of the PREMIS objects. Further alignment with OAIS requirements as expressed in PREMIS and other preservation will take place as DDI-L expands into process models, provenance, and archive management content.

3.0 General Structures

3.1 - Identification, Versioning, Maintenance and Reference

The identification structure of DDI objects is core to the functioning of the standard. The purpose of the DDI identification structure is to:

- Uniquely identify major objects in a persistent manner to ensure accurate reference and retrieval of the object content
- Provide context for objects where an understanding of related object types within a packaging structure is essential to the understanding of the object (i.e., a specific classification within a classification scheme)
- Manage metadata object change over time
- Support the range of object management used by different organizations

- Support early and late binding of references
- Support interaction with closely related standards, in particular ISO/IEC 11179 and SDMX

The identification structure is based on the ISO/IEC 11179 structure that requires a three-part means of unique identification.

OBJECT	ISO/IEC 11179	DDI
Agency Identifier	A unique identifier for the agency managing the object	A unique identifier for an agency registered with the DDI Alliance. The agency may have multiple sub-agency extensions managed within the DDI Registry or within the primary agency. An agency and sub-agency are separated by an ".".
Unique ID	A unique identifier of the object within the context of the agency	An identification which is unique within a) the agency (sub-agency), or b) within the parent maintainable. If the context is the parent maintainable the Unique ID is the ID of the parent maintainable plus the ID of the object within that maintainable separated by a ".".
Version Number	A version number of the object to track change over time	A version number with any number of extensions separated by a ".".

This three part structure is the equivalent of a unique persistent identifier for an object, such as described by DOIs and other similar structures. Note that while use of a version number with a DOI is optional, based on local practice, the Version Number in DDI is required due to the need to manage metadata within a dynamic workflow over time.

Identifiable, Versionable, and Maintainable

DDI differentiates between a set of element types. Not all objects are individually identifiable, i.e. some objects only have meaning within the context of an identifiable object such as a Label or Description. The remaining objects are Identifiable, Versionable or Maintainable in order to support different levels of metadata management. Identifiable objects are those that can be referenced directly either for inclusion in another object or for the purpose of attaching Other Material or a Note to the object.

Identifiable objects have a unique ID within the context of their specified scope of uniqueness (see Scope of Uniqueness discussion within this section). Their Agency Identification and Version Number match those of the object's immediate parent Versionable (or Maintainable if there is not a parent Versionable) at the ***point of creation***. This means that if an Identifiable object is created within a version

1 of a parent Versionable and does NOT change its content over time it will retain its Version Number of 1 until the identifiable object itself is altered. It will then change its Version Number to that of its parent Versionable ***at the time of the alteration.*** In other words an Identifiable could go from a Version 1 to a Version 4 without ever having a Version 2 or 3 if the cause for versioning did not involve any change in the Identifiable object within Version 2 and 3 of the parent Versionable.

A *Versionable object* has the characteristics of an Identifiable object but may be managed over time. DDI has determined that being able to track change within the object over time is a requirement, either to understand the relationship to earlier objects of a similar type or to track provenance. Note that it is up to the individual content provider to determine whether an object is essentially new or is a modification (version) of an earlier object. Versionable objects have a unique ID within the context of their specified scope of uniqueness (see Scope of Uniqueness discussion within this section). Their Agency identification matches that of the object's immediate parent Maintainable at the ***point of creation***. In other words the Agency of an object does not change simply because it is included by reference in a Maintainable managed by a different agency. The Version number of the object changes each time its content changes. See Versioning for a discussion of when and how this may be implanted within different organizations or projects.

A *Maintainable object* is a type of packaging and generally takes the form of either a module or scheme. Modules package metadata focused on specified segments of the Lifecycle for which context is important for understanding. Schemes are similar to data base tables, containing a stack of similar type objects that many have important contextual relevance to each other, i.e. a classification scheme captured in a DDI Category Scheme. There is one unique form of a Maintainable which is the CodeList. A CodeList is a Maintainable in its own right for the purpose of supporting the statistical production process. However it can only be published within the context of a parent Maintainable Scheme. All Schemes and Modules may be published within the context of a Study Unit or Group (a collection of Study Units) or as a separate Resource Package item primarily for the purpose of reuse.

Versioning

In general, once metadata is published (i.e. flagged as being stable for referencing purposes) any change to the content should result in a version change to the Versionable object containing the change. Note that as Versionable objects are contained by Maintainable objects and that Maintainable object may be contained in another Maintainable object, a single change will generally trigger Versioning up the containing tree of the metadata. The purpose of versioning is to ensure that someone referencing a specific version of an object will always get back the same information.

Versioning rules and Version Number structures differ between different organizations and between different projects or products of the same organization. DDI does not dictate this structure EXCEPT to specify that it must be expressed as one or more integers separated by a "." (dot). The DDI Lifecycle specification uses a three part structure of a Major.Minor.Sub-Minor version number. The Organization should describe its versioning system or systems so that it is clear to the user when and how versioning occurs. Versioning is "required" once a Maintainable object is flagged isPublished="true". During

production processes, tracking version changes may be managed by other means such as Version Date or an external subversion system.

Some organizations are stricter in their versioning rules than others. For example, a typographical correction within a Description text which is considered to have no impact on the intellectual content may trigger a Minor or Sub-minor version change in one system while only result in Version Date change in another. Because the impact of a change cannot be easily predicted (it is dependent upon the use of the metadata) the important thing is to capture that a change was made and to provide the end user with a clear set of versioning rules that supports their ability to evaluate the impact of the change for their particular use. In addition, some metadata is considered local in nature, specific to interaction with an identified system. This metadata is considered to be Administrative in nature and is viewed by DDI as a set of metadata that does not alter the intellection content (Payload) of the metadata and does not need to drive a version change.

Administrative and Payload Metadata

The differentiation between Administrative metadata and Payload metadata lies primarily in how changes to the metadata content drive version changes.

Administrative metadata is content that is related to the interaction or use of the metadata within a specific system. Changes in administrative metadata do not change the meaning of the metadata content of the object in terms of its DDI identification or intellectual meaning. For example, the addition of a local User Identification to facilitate interaction between the metadata object and local access software, or creating a URN entry from existing identification sequence content are considered to be changes to the Administrative metadata. A change in Administrative metadata does not trigger a change in the version number of a published object.

Payload metadata is all metadata NOT defined as Administrative. Payload metadata contains intellectual content that has an impact on the meaning or application of the object. A change in Payload metadata generally triggers a change in the version number of a published object.

The following objects which consist of the extension bases used for identification and referencing purposes are considered to be Administrative metadata for the purposes of versioning:

Extension bases are listed in RED and contained objects in BLACK	***General description***
IDENTIFIABLE OBJECT	Extension base is AbstractIdentifiable
URN @typeOfIdentifier Agency ID Version UserID @typeOfUserID @userIDVersion @typeOfUserVersion MaintainableObject TypeOfObject MaintainableID MaintainableVersion @inheritanceAction @objectSource @scopeOfUniqueness @isIdentifiable	See URN Structure below URN used by Agency [Canonical | Deprecated] Base sequence of identification. If sequence is used, all are required. A user specified identification for a specific system such as a DOI or internal search engine. Both the ID and @typeOfUserID must be specified. Version information is optional. The Maintainable object containing the Identifiable object. TypeOfObject is required to create Deprecated URN. ID is required for any URN if ScopeOfUniqueness=”Maintainable”. Version number provides context base. See Grouping: Inheritance The object source of a resolved reference Scope used by Agency [Agency | Maintainable Fixed value = “true” [specifies object base]
VERSONABLE OBJECT	Extension base is AbstractVersionable
URN @typeOfIdentifier Agency ID Version UserID @userIDVersion @typeOfUserVersion UserAttributePair AttributeKey AttributeValue VersionResponsibility VesionResponsibilityReference VersionRationale BasedOnReference MaintainableObject TypeOfObject MaintainableID	Same as Identifiable Object User defined Key/Value pair used to support interaction of the metadata within the user’s system. Who within the Agency versioned the object Alternate reference to who versioned the object Reason for version change (to inform user) Intellectual base of this object (local version of an external object) Same as Identifiable Object

MaintainableVersion @typeOfIdentifier @inheritanceAction @objectSource @scopeOfUniqueness @isVersionable @versionDate	 Fixed value = “true” [specifies object base] Date/Time of version change
MAINTAINABLE OBJECT	Extension base is AbstractMaintainable
URN @typeOfIdentifier Agency ID Version UserID @typeOfUserID @userIDVersion @typeOfUserVersion UserAttributePair AttributeKey AttributeValue VersionResponsibility VesionResponsibilityReference VersionRationale BasedOnReference Note Software MetadataQuality @inheritanceAction @objectSource @scopeOfUniqueness @isMaintainable @versionDate @externalReferenceDefaultURI @isPublished @xml:lang	Same as Versionable object Notes related to objects with the maintainable (Payload) Software used to create the Maintainable object (Payload) Quality of the metadata in the Maintainable object (Payload) Same as Versionable object Fixed value = “true” [specifies object base] Same as Versionable object Indicates that the content is available for reuse by reference The language of the metadata in the Maintainable object (Payload)
REFERENCE TYPE URN	 The URN of the object being referenced using

	the specified URN structure
@typeOfIdentifier	URN used by Agency [Canonical | Deprecated]
Agency	The agency of the object being referenced.
ID	The ID of the object being referenced.
Version	The Version of the object being referenced. This is the full Version number at the time the reference is created.
TypeOfObject	Type of object being referenced. This is a controlled list.
MaintainableObject TypeOfObject MaintainableID MaintainableVersion	The Maintainable object containing of the Identifiable or Versionable object being referenced. TypeOfObject is required to create Deprecated URN. ID is required for any URN if ScopeOfUniqueness="Maintainable". Version number provides context base.
@isExternal	Boolean attribute. If "true" you must supply the URN
@externalReferenceDefaultURI	A local store for the referenced object expressed as a URI
@isReference	Fixed value = "true" [specifies object base]
@lateBound	Boolean attribute. Set to "true" if you wish to late-bind the reference (i.e., want to reference the most recent version)
@lateBoundRestriction	If @lateBound="true" and this attribute is not provided you will get the most recent version. Use this attribute to restrict the value of the late-bind, for example to restrict to the most recent minor version of major version 4 @lateBoundRestriction="4".
@objectLanguage	Specify the desired language content (if available) for the referenced item using a valid xml:lang value.
@sourceContext	The URN of the parent maintainable at the time of reference (this is not necessarily the same version number as the version of the parent maintainable at point of origin)
SCHEME REFERENCE TYPE	
URN @typeOfIdentifier Agency ID Version TypeOfObject MaintainableObject TypeOfObject	Same as Reference.

MaintainableID MaintainableVersion Exclude ID Version @isExternal @externalReferenceDefaultURI @isReference @lateBound @lateBoundRestriction @objectLanguage @sourceContext	 Allows the identification of objects within the Scheme which are to be excluded for the purpose of this reference. The ID of the excluded object. The Version of the excluded object. Same as Reference.

Scope of Uniqueness

DDI 3.2 supports scoping the uniqueness of identifier to the parent Maintainable or to the Agency (sub-agency). This choice affects the structure of the ID as it appears within the Canonical URN (see Type of Identifier in this section). Essentially, when the ID is scoped to the Agency the unique identification of an object requires the Agency, ID of the object, and Version Number of the object. When the ID is scoped to the Maintainable the unique identification of a non-Maintainable object requires the Agency, ID of the parent Maintainable, the ID of the object, and the Version Number of the object. The attribute scopeOfUniqueness is required and must contain either “Agency” or “Maintainable”. This attribute defines how the ID will be expressed in the Canonical URN and what is required for a complete reference to the object within the Maintaining Agency.

Type of Identifier

DDI 3.2 supports two formats for expressing the Identification Sequence as a URN. The Canonical URN is recommended. The Deprecated URN is a modification of the DDI 3.1 URN. Whether the identification information is expressed as a URN or using the Identification Sequence the attribute “typeOfIdentifier” on the URN dictates the format of the URN supported internally by the agency and the format of the URN used by an external reference to that object. Note that regardless of the type of identifier used it is good practice to provide the full content of the Identification Sequence and details of the MaintainableObject for non-Maintainable objects in order to support the creation of any format of a DDI URN.

Structure of the URN

Canonical URN

Each section of the Canonical URN is separated by a “:” (colon). Within the ID section the Maintainable ID and Object ID are separated by a “.” (dot).

“urn:ddi:agency[.sub-agency]:ID:Version”

If the scopeOfUniqueness equals “Agency” the ID is the ID of the object.

Example:

Canonical URN with uniqueness scoped to the Agency.

Object V321 version 2 within the Minnesota Population Center (us.mpc)

urn:ddi:us.mpc:V321:2

Object V321 version 2 within the Minnesota Population Center, Project IPUMS (listed as a sub-agency within us.mpc)

urn:ddi:us.mpc.ipums:V321:2

If the scopeOfUniqueness equals “Maintainable” the ID of a non-Maintainable object is structured as follows:

“urn:ddi:agency[.sub-agency]:MaintainableID.ObjectID:Version”

Canonical URN with uniqueness scoped to the Maintainable.

Variable V321 version 2 within VariableScheme VS1 at the Minnesota Population Center (us.mpc)

urn:ddi:us.mpc:VS1.V321:2

Variable V321 version 2 within VariableScheme VS1 at the Minnesota Population Center, Project IPUMS (listed as a sub-agency within us.mpc)

urn:ddi:us.mpc.ipums:VS1.V321:2

Deprecated URN

Each section of the Deprecated URN is separated by a “:” (colon).

“urn:ddi:agency[.sub-agency]:MaintainableObjectType:MaintainableID:ObjectType:ObjectID:ObjectVersion”

If the object itself is Maintainable the information on the parent maintainable is not included:

“urn:ddi:agency[.sub-agency]: ObjectType:ObjectID:ObjectVersion”

If the scopeOfUniqueness equals “Agency” the ID is the ID of the object.

Example:

Deprecated URN with uniqueness scoped to the Agency.

Object V321 version 2 within the Minnesota Population Center (us.mpc)

```
urn:ddi:us.mpc:Variable:V321:2
```

Object V321 version 2 within the Minnesota Population Center, Project IPUMS (listed as a sub-agency within us.mpc)

```
urn:ddi:us.mpc.ipums:Variable:V321:2
```

If the scopeOfUniqueness equals “Maintainable” the ID of a non-Maintainable object is structured as follows:

“urn:ddi:agency[.sub-agency]:MaintainableObjectType:MaintainableID:ObjectType:ObjectID:ObjectVersion”

Deprecated URN with uniqueness scoped to the Maintainable.

Variable V321 version 2 within VariableScheme VS1 at the Minnesota Population Center (us.mpc)

```
urn:ddi:us.mpc:VariableScheme:VS1:Variable:V321:2
```

Variable V321 version 2 within VariableScheme VS1 at the Minnesota Population Center, Project IPUMS (listed as a sub-agency within us.mpc)

```
urn:ddi:us.mpc.ipums:VariableScheme:VS1:Variable:V321:2
```

EXAMPLES:

Note that by including the information for the MaintainableObject in an Identifiable or Versionable the user has provided sufficient information to build either a Canonical or Deprecated URN scoped to either the agency or the maintainable. While the information may not be part of the URN as expressed by the maintaining agency, the information contained in the MaintainableObject provides contextual information that may be important to the user and may need to be passed to them for purposes other than strict identification. The inclusion of Versioning information in Versionable and Maintainable may be used internally to track quality control and processing activities and may also be valuable to the user in determining whether the change caused by versioning will affect their analysis work.

Identifiable

```
<l:Code isIdentifiable="true" scopeOfUniqueness="Maintainable">
   <r:URN typeOfIdentifier="Canonical">
      urn:ddi:us.mpc:CL_1.Code_1:1
   </r:URN>
   <r:Agency>us.mpc</r:Agency>
   <r:ID>Code_1</r:ID>
   <r:Version>1</r:Version>
   <r:UserID typeOfUserID="NHGIS">Gender_1</r:UserID>
   <r:MaintainableObject>
      <r:TypeOfObject>CodeList</r:TypeOfObject>
      <r:MaintainableID>CL_1</r:MaintainableID>
      <r:MaintainableVersion>1</r:MaintainableVersion>
   </r:MaintainableObject>
   ...
```

```
</l:Code>

Minimum required content of the above:

<l:Code isIdentifiable="true" scopeOfUniqueness="Maintainable">
   <r:URN typeOfIdentifier="Canonical">
      urn:ddi:us.mpc:CL_1.Code_1:1
   </r:URN>
   ...
</l:Code>
```

Versionable

```
<l:Variable isVersionable="true" scopeOfUniqueness="Agency"
versionDate="2012-10-31">
   <r:URN typeOfIdentifier="Canonical">
      urn:ddi:us.mpc:Var_1234:2
   </r:URN>
   <r:Agency>us.mpc</r:Agency>
   <r:ID>Var_1234</r:ID>
   <r:Version>2</r:Version>
   <r:UserID typeOfUserID="IPUMS">MOMLOC</r:UserID>
   <r:VersionResponsibility>John Doe</r:VersionResponsibility>
   <r:VersionRationale>
      <r:RationaleDescription><r:String xml:lang="en">Expanded
description to more clearly define the process of determining the
MOCLOC value in households with multiple
mothers.</r:String></r:RationaleDescription>
      <r:RationaleCode>Update</r:RationaleCode>
   </r:VersionRationale>
   <r:MaintainableObject>
      <r:TypeOfObject>VariableScheme</r:TypeOfObject>
      <r:MaintainableID>VS_IPUMS</r:MaintainableID>
      <r:MaintainableVersion>6</r:MaintainableVersion>
   </r:MaintainableObject>
   ...
</l:Variable>

Minimum required content of the above:

<l:Variable isVersionable="true" scopeOfUniqueness="Agency"
versionDate="2012-10-31">
   <r:URN typeOfIdentifier="Canonical">
      urn:ddi:us.mpc:Var_1234:2
   </r:URN>
   ...
</l:Variable>
```

Maintainable

```
<l:VariableScheme isVersionable="true" scopeOfUniqueness="Agency"
versionDate="2012-10-31" isPublished="true" xml:lang="en">
```

```
   <r:URN typeOfIdentifier="Canonical">
      urn:ddi:us.mpc:VS_IPUMS:6
   </r:URN>
   <r:Agency>us.mpc</r:Agency>
   <r:ID>VS_IPUMS</r:ID>
   <r:Version>6</r:Version>
   <r:UserID typeOfUserID="IPUMS">IPUMS_VARS</r:UserID>
   <r:VersionResponsibility>John Doe</r:VersionResponsibility>
   <r:VersionRationale>
      <r:RationaleDescription>
         <r:String xml:lang="en">Versioned MOMLOC</r:String>
      </r:RationaleDescription>
      <r:RationaleCode>Update</r:RationaleCode>
   </r:VersionRationale>
   <r:Software></r:Software>
   <r:MetadataQuality>
      <r:QualityMeature>InternalReview</r:QualityMeasure>
      <r:MeasurePurpose><r:Content xml:lang="en">Content has be
reviewed and confirmed for use on Public
site.</r:Content></r:MeasurePurpose>
      <r:MeasureValue>Public</r:MeasureValue>
   </r:MetadataQuality>
   ...
</l:VariableScheme>

Minimum required content of the above:

<l:VariableScheme isVersionable="true" scopeOfUniqueness="Agency"
versionDate="2012-10-31" isPublished="true" xml:lang="en">
   <r:URN typeOfIdentifier="Canonical">
      urn:ddi:us.mpc:VS_IPUMS:6
   </r:URN>
   ...
</l:VariableScheme>
```

Reference

Note that in this case Version 1 of Var_1234 originally appeared in Version 1 of VS_IPUMS. However, the sourceContext indicates that VS_IPUMS:4 is the context at the point of reference.

```
<l:VariableReference isReference="true" isExternal="false"
lateBound="false" objectLanguage="en"
sourceContext="urn:ddi:us.mpc:VS_IPUMS:4.0">
   <r:URN typeOfIdentifier="Canonical">
      urn:ddi:us.mpc:Var_1234:1.0
   </r:URN>
   <r:Agency>us.mpc</r:Agency>
   <r:ID>Var_1234</r:ID>
   <r:Version>1.0</r:Version>
   <r:TypeOfObject>Variable</r:TypeOfObject>
   <r:MaintainableObject>
```

```
      <r:TypeOfObject>VariableScheme</r:TypeOfObject>
      <r:MaintainableID>VS_IPUMS</r:MaintainableID>
      <r:MaintainableVersion>1.0</r:MaintainableVersion>
   </r:MaintainableObject>
</l:VariableReference>
```

Minimum required content of the above:

```
<l:VariableReference isReference="true" isExternal="false"
lateBound="false">
   <r:URN typeOfIdentifier="Canonical">
      urn:ddi:us.mpc:Var_1234:1.0
   </r:URN>
   <r:TypeOfObject>Variable</r:TypeOfObject>
</l:VariableReference>
```

Above reference as a lateBound reference where the most recent minor version of major version 1 of the variable is being requested.

```
<l:VariableReference isReference="true" isExternal="false"
lateBound="true" objectLanguage="en"
sourceContext="urn:ddi:us.mpc:VS_IPUMS:4.0" lateBoundRestriction="1">
   <r:URN typeOfIdentifier="Canonical">
      urn:ddi:us.mpc:Var_1234:1.0
   </r:URN>
   <r:Agency>us.mpc</r:Agency>
   <r:ID>Var_1234</r:ID>
   <r:Version>1.0</r:Version>
   <r:TypeOfObject>Variable</r:TypeOfObject>
   <r:MaintainableObject>
      <r:TypeOfObject>VariableScheme</r:TypeOfObject>
      <r:MaintainableID>VS_IPUMS</r:MaintainableID>
      <r:MaintainableVersion>1.0</r:MaintainableVersion>
   </r:MaintainableObject>
</l:VariableReference>
```

SchemeReference

```
<l:VariableSchemeReference isReference="true" isExternal="false"
lateBound="false" objectLanguage="en">
   <r:URN typeOfIdentifier="Canonical">
      urn:ddi:us.mpc:VS_IPUMS:1.0
   </r:URN>
   <r:Agency>us.mpc</r:Agency>
   <r:ID>VS_IPUMS</r:ID>
   <r:Version>1.0</r:Version>
   <r:TypeOfObject>VariableScheme</r:TypeOfObject>
   <r:Exclude isReference="true" isExternal="false" lateBound="false"
typeOfIdentifier="Canonical">
      <r:URN>urn:ddi:us.mpc:Var_1234:1.0</r:URN>
      <r:TypeOfObject>Variable</r:TypeOfObject>
   </l:Exclude>
```

```
</l:VariableSchemeReference>

Minimum required content of the above:

<l:VariableSchemeReference isReference="true" isExternal="false"
lateBound="false" objectLanguage="en">
   <r:URN typeOfIdentifier="Canonical">
      urn:ddi:us.mpc:VS_IPUMS:1.0
   </r:URN>
   <r:TypeOfObject>VariableScheme</r:TypeOfObject>
   <r:Exclude isReference="true" isExternal="false" lateBound="false"
typeOfIdentifier="Canonical">
      <r:URN>urn:ddi:us.mpc:Var_1234:1.0</r:URN>
      <r:TypeOfObject>Variable</r:TypeOfObject>
   </l:Exclude>
</l:VariableSchemeReference>
```

3.2 - Maintainable Objects: Modules and Schemes

Maintainable objects are of two basic types, Modules and Schemes. Modules roughly relate to stages in the life cycle or publishing packages. Publishing packages pull together related material regarding a study, a series or group of studies, reusable resources, or deposited objects. Modules like DataCollection, LogicalProduct, PhysicalDataProduct, and ConceptualComponents bring together sets of information related to specific activities, the results of a study, or conceptual background. They may include other modules, schemes, or non-scheme based objects.

Schemes are designed to bring together sets of similar objects (i.e. Variables, Concepts, RecordLayouts, etc.) which are managed as a unit for either conceptual or administrative purposes. Schemes have a common structure providing descriptive information about the scheme, a set of similar objects, and the ability to group those objects for administrative purposes.

3.3 – Name, Label, and Description

DDI modules designed to contain published objects (DDIInstance, StudyUnit, Group, ResourcePackage, LocalHoldingPackage, and PhysicalInstance) all contain full citation information which provides detailed information on the name of the object, as well as means of labeling and describing it. DDI has identified a number of objects, primarily non-publication structure modules, schemes, and versionable objects within schemes as items that have the potential to be managed in an ISO/IEC 11179 type repository. In line with ISO/IEC 11179-5, DDI has provided this set of objects with a sequence of a name, label, and description. Label and Description are standard structures. Name is used as a type specification for a specific object name, i.e. VariableName type=”r:NameType”. A complete listing of objects of NameType will be found in Appendix: X.

NameType	
Namespace: **r** *Parent Maintainable:* **varies by usage**	**InternationalStringType**
Required Referenced Objects:	*Optionally Referenced Objects:*

NameType	
String	(0..n)
@xml:lang	(optional)
@isTranslated	(default=”false”)
@isTranslatable	(default=”true”)
@translationSourceLangauge	(optional)
@translationDate	(optional)
@isPreferred	(optional)
@context	(optional)

A name is what an object is called within a registry. All elements of type NameType may be repeated to supply different names for different contexts, such as different sections within a registry system. Do not repeat the use of the NameType object to capture multilingual content. This is handled by the base type, InternationalString [see pt2:2.5.1]. Each String within the NameType represents the same content in a different language. Each language string can be identified by its language designation (with optional country specification, i.e. en-UK), with information on whether the content was translated, can be translated (i.e. is not machine code), the source langauges for the translation, and the translation date. The two attributes specific to NameType identify the preferred name if multiple name sets are available and capture the context within which the specified name is used. If the element of type NameType is repeated, it is the attribute context which is used to disambiguate between the options. The attribute isPreferred allows the content creator to designate the preferred or default name for the object.

Label	
Namespace: **r** *Parent Maintainable:* **varies by usage**	**StructuredStringType**
Required Referenced Objects:	*Optionally Referenced Objects:*
Label	
Content	(0..n)
xhtml:BlkNoForm.mix	(0..n)
@xml:lang	(optional)
@isTranslated	(default=”false”)
@isTranslatable	(default=”true”)
@translationSourceLangauge	(optional)
@translationDate	(optional)
TypeOfLabel	(0..n)
@locationVariant	(optional)
@validForStartDate	(optional)
@validForEndDate	(optional)
@maxLength	(optional)

A Label is intended to provide content for use in a display (a table layout, printed content, web site, etc.). In all locations where Label is used it may be repeated to reflect different types of label (i.e. a short label, or a full label, etc.). The Label uses an extension base of StructuredStringType which managed both multilingual content and allows for the use of a set of xhtml structure tags (see pt2:2.5.1). Differences in the intended use of each Label are expressed by the element TypeOfLabel and a set of attributes that apply to all language versions of the Label. The TypeOfLabel uses the extension base CodeValueType which supports the use of an external controlled vocabulary (see pt2:2.5.1). The attribute locationVariant is an xs:string to allow for either a common geographic specification such as a country code or a descriptive term (i.e. urban, northwest region, etc.). For Labels with a limited period of use the pair of attributes validForStartDate and validForEndDate allow clear specification for when a Label is valid. Finally the attribute maxLength is useful for identifying labels that must meet a specific length limitation, for example in publishing content within a specific format or software system.

Description	
Namespace: **r** *Parent Maintainable:* **varies by usage**	**StructuredStringType**
Required Referenced Objects:	*Optionally Referenced Objects:*
Description	
Content	(0..n)
xhtml:BlkNoForm.mix	(0..n)
@xml:lang	(optional)
@isTranslated	(default="false")
@isTranslatable	(default="true")
@translationSourceLangauge	(optional)
@translationDate	(optional)

Description is of type StructuredStringType with no additional extensions. It generally appears with a cardinality of 0..1. If it is important to differentiate be original and added content it is possible to do this by using the content structure features of the StructuredString (see pt2:2.5.1). Note that if the object containing the Name, Label, Description sequence may be used in a comparison process, providing content for Description is critical as comparison is based upon the comparison of the Description of the object, not its Name or Label.

3.4 - Controlled Vocabularies

There are many points in the DDI where a controlled vocabulary is desired, but no single classification can be (or has been) identified which would be acceptable to all user communities. DDI-L provides a CodeValueType that allows for use of a simple descriptive term while also supporting the use of an externally described controlled vocabulary. A set of fields has been made available for identifying the following information about the controlled vocabulary including; the name or identifier, maintenance agency, version, and URL of the controlled vocabulary. DDI is supporting the option of developing and publishing controlled vocabularies expressed in a DDI profile of Genericode. These may be used directly

or incorporated into local publications of controlled vocabularies that reflect those elements that are common with the DDI community and adding those that are specific to the the maintenance agency. This approach supports sharing of common coding structures as well as the publication of code schemes in formats that can be mapped for comparability. DDI publishes a set of controlled vocabularies commonly used in social science research (http://www.ddialliance.org/controlled-vocabularies). These are presented in a variety of formats including Genericode XML.

4.0 - Specific Structures in Alphabetical Order:

4.1 - Archive

When DDI talks of archive it is referring to specific activities that take place for the purpose of organizing and preserving the metadata and related data files. Archive activities are performed by the individual or organization responsible for maintaining and preserving the metadata. This could be the individual researcher, a data production organization, or a formal archive dedicated to the preservation of metadata and data. It focuses on the information related to management, preservation, and access.

Archive is a module which contains two types of information. The first is information specific to the instance of the metadata in a specific location. This "ArchiveSpecific" information may have no relevance outside of a specific location. For example, information on the physical storage location of the metadata parts, how they fit into a local collection, or what related materials the archive may contain. In addition to the "ArchiveSpecific" information, the Archive module contains a record of events that are significant in the life of the metadata and houses the OrganizationScheme which describes the organizations and individuals related to the metadata. OrganizationScheme and its contents are described in "Organizations, Individuals, and Relations" (xx.xx.xx). An OrganizationScheme may be included in-line or by reference. This allows for the maintenance of a master OrganizationScheme as a ResourcePackage and the inclusion of the relevant sections in individual Archive modules.

The Archive section describes the overall structure of the module, describes the ArchiveSpecific information and provides details on AccessType which is used to describe both default and specific access restrictions at the level of the archive, the collection, or an item.

```
Archive
  Extension base: r:MaintainableType
  ArchiveModuleName                              (0..n)
  r:Label                                        (0..n)
  r:Description                                  (0..1)
  ArchiveSpecific                                (0..n)
  CHOICE                                         (0..n)
       OrganizationScheme
       r:OrganizationSchemeReference
  END CHOICE
  r:LifecycleInformation                         (0..1)
       LifecycleEvent                            (0..n)
           Extension base: IdentifiableType
           Label                                 (0..n)
```

```
          EventType                                (0..1)
          Date                                     (0..1)
          AgencyOrganizationReference              (0..n)
          Description                              (0..1)
          Relationship                             (0..n)
  r:OtherMaterial                                  (0..n)
```

The Archive contains an ArchiveModuleName, Label, and Description to support its maintenance within a registry system. The purpose of the module is to organize the information about the location, management, and important events surrounding the metadata to which it belongs. ArchiveSpecific information is discussed below and the details of the OrganizationScheme are found in “Organizations, Individuals, and Relations” (xx.xx) . A key feature in Archive is the LifeCycleInformation object which contains an unsorted stack of LifecycleEvents. These events are identifiable and may have Note or OtherMaterials attached to them. These should all be contained within the Archive module so they do not risk becoming disassociated with the related object.

A LifecycleEvent is any event that the user decides has value to the archive or the end user. Capturing data and metadata processing information related to ingest and management of metadata within an archive or similar organization is a common use of this section. The LifecycleEvent contains a label for the event, specifies the type of event being captured (supports the use of a controlled vocabulary for this object), specifies the date of the event, the organizations or individuals involved by reference to their description in an OrganizationScheme, and describes the event. The event may be attached to one or more identifiable objects within the metadata by use of Relationship. This is the same structure as used in Note(xxx.xx) and OtherMaterial (xxx.xx)

```
ArchiveSpecific
     ArchiveOrganizationReference                (0..1)
     Item                                        (0..n)
          r:Citation                             (0..1)
          LocationInArchive                      (0..n)
          CallNumber                             (0..1)
          r:URI                                  (0..1)
          ItemFormat                             (0..1)
          Media                                  (0..1)
          StudyClass                             (0..1)
               r:Description                     (0..1)
               ClassType                         (0..1)
          Access                                 (0..1)
          OriginalArchiveOrganizationReference   (0..n)
          AvailabilityStatus                     (0..1)
          DataFileQuantity                       (0..1)
          CollectionCompleteness                 (0..1)
          Item                                   (0..n)
     Collection                                  (0..n)
          r:Citation                             (0..1)
          LocationInArchive                      (0..n)
          CallNumber                             (0..1)
```

```
        r:URI                                   (0..1)
        ItemQuantity                            (0..1)
        StudyClass                              (0..1)
             r:Description                      (0..1)
             ClassType                          (0..1)
        DefaultAccess                           (0..1)
        OriginalArchiveOrganizationReference    (0..n)
        AvailabilityStatus                      (0..1)
        DataFileQuantity                        (0..1)
        CollectionCompleteness                  (0..1)
        Item                                    (0..n)
        Collection                              (0..n)
   DefaultAccess                                (0..n)
   r:FundingInformation                         (0..n)
   r:Budget                                     (0..n)
   r:QualityStatementReference                  (0..n)
   r:Coverage                                   (0..1)
```

ArchiveSpecific focuses on the place of the metadata within the collections of the organization which maintains the original or a local depository copy. The first object in ArchiveSpecific is a reference to the description of the ArchiveOrganization itself. This description within an OrganizationScheme should contain all of the relevant contact information for the Archive Organization, any special identifiers such as its DDI Agency identification, information on how it handles versioning, and any other information that would be useful to anyone needing to interact with the agency. Item and Collection, both of which can internally nested objects of the same type, provide flexibility for the archive to describe its holdings. At the level of ArchiveSpecific you can describe the general or DefaultAccess rules and processes for the Archive (these may be superseded by rules for specific items or collections), FundingInformation for archival processing or collection and Budget information. QualityStatements, such as adherence to OAIS or various certification programs, are included here by reference. A statement of the overall coverage of the archive may be described in spatial, temporal, and topical terms.

Item and Collection are similar in structure. Archives deal with addressable byte streams, files, representations (objects collectively representing an intellectual entity), as well as collections and sub-collections of those objects. An archive should be internally consistent in how it organizes this information about the objects it contains, but DDI does not dictate the organization structure. For example, a Representation may be described as an Item containing several sub-items each representing a file and/or byte stream. Others may define a Representation a form of Collection with an Item used to define individual files and byte streams.

Both Item and Collection capture the following information about a digital object: a citation for the object, it's physical location in the archive such as a remote or offline storage location, a CallNumber (internal identification number), the URI, the StudyClass which is used to represent any processing or other non-topical means of classification within the archive, a reference to the original archive organization (where did it come from), availability status, quantity of related data files, a statement concerning the completeness of the collection, and a recursive object (Item or Collection) used to create nested hierarchies.

The unique objects in Item refer to the physical manifestation of a single item including the ItemFormat, Media, and Access restrictions/permissions specific to the Item. The unique objects in Collection reflect its description of a set of objects including, ItemQuantity (a check sum), and DefaultAccess providing restrictions/permissions for the set (these can be overridden at the item level).

```
AccessType
      Extension base: r:IdentifiableType
      AccessTypeName                              (0..n)
      r:Label                                     (0..n)
      r:Description                               (0..1)
      ConfidentialityStatement                    (0..1)
      AccessPermission                            (0..n)
            FormNumber                            (0..1)
            r:URI                                 (0..1)
            Statement                             (0..1)
            @isRequired                           (optional)
      Restrictions                                (0..1)
      CitationRequirement                         (0..1)
      DepositRequirement                          (0..1)
      AccessConditions                            (0..1)
      Disclaimer                                  (0..1)
      AccessRestrictionDate                       (0..1)
      ContactOrganizationReference                (0..n)
```

AccessType is used by both Access and DefaultAccess to describe the details of any permissions or restrictions related to a specific object or a set of objects. In general, default access specifications at the archive or collection level are applied to their contained objects unless overridden by conflicting Access information lower down. For example, an archive may have a policy of open access to all users, but a specific collection may require registration, or an individual object may be private and inaccessible to any users outside of the research project that created it.

AccessType is identifiable so that external documents may be associated with it. It contains a Name, Label, and Description to assist in managing multiple access statements. Note that while the term "restriction" is used the content may be captured as descriptions of specific restrictions or permissions, whichever is clearer. AccessRestrictionDate provides a timeframe for the access description. ContactOrganizationReference provides a reference to the organization or individual to contact for further information regarding access. If an organization or sub-organization is referenced, changes in personnel or contact numbers can be captured in the OrganizationScheme. The following descriptive content are all of type StructuredStringType supporting both language replication and structured content for clarity: ConfidentialityStatement, Restrictions, CitationRequirement, DepositRequirement, AccessConditions, and Disclaimer. AccessPermission is a reference to a specific form, either physical or digital, used to arrange for access to an item. It contains a form number, a URI, a Statement relaying the use and purpose of the access form (InternationalStringType), and a Boolean attribute indicating whether completion of the form is required for access.

The following example contains all the major sections noted above including the ArchiveSpecific information, a short set of Items within a collection, three LifecycleEvents, and a related OtherMaterial brief citation. Access information is provided for the Archive (MPC_NHGIS) as DefaultAccess.

Archive Example (ArchiveExample.xml)

4.2 - Date

With the exception of describing the date format in a data capture system or as it is represented in a data file, DDI requires the use of the standard ISO 860 formats for all machine actionable dates. ISO 860 uses the Gregorian calendar so dates in other calendars must be translated. The basic form of a date in DDI is the BaseDateType which is a union of ISO dates types including:

xs:dateTime	yyyy-mm-ddThh:mm:ss	1982-01-05T23:05:15
xs:date	yyyy-mm-dd	1982-01-05
xs:gYearMonth	yyyy-mm	1982-01
xs:gYear	yyyy	1982
xs:duration	PnnYnnMnnDTnnHnnMnnS	P26Y02M22DT11H05M20S

Note that all upper case letters are literals, for example, the "T" in dateTime is literal, denoting the beginning of the Time section. Seconds (ss) may contain decimals. Optionally, dateTime can be extended by a time zone offset of "Z" to represent Zulu time or GMT. For example, Eastern Standard Time is Z-4, Central Europe is Z+1.

"P" in xs:duration indicates that this is a Period of duration and the number precedes the type of period, (i.e., nnY is the number of Years). A period may be of negative duration by placing a negative sign before the "P", for example a period of minus 10 days (-P10D).

Elements of DateType support the use of a single date or date range. Date ranges are assumed to be inclusive. While normally a range will have a StartDate and EndDate, DDI supports the use of open ranges where only the StartDate or EndDate is known. SimpleDate, StartDate, and EndDate are all BaseDateType.

All dates may be replicated in their HistoricalDate format to reflect how the date was expressed in original documentation. Historical date information parallels the simple date, start date and end date structures of the standard DateType. The date is captured as a string in NonISODate, the format is provided in HistoricalDateFormat (supports the use of a controlled vocabulary), and the calendar type may be specified in Calendar (supports the use of a controlled vocabulary). HistoricalDate, HistoricalStartDate, and HistoricalEndDate are all HistoricalDateType

The overall date format is composed of three primary structures, BaseDateType, DateType, and HistoricalDateType:

```
BaseDateType
      Union: xs:dateTime xs:date xs:gYearMonth xs:gYear xs:duration
```

```
DateType
      CHOICE:
            SEQUENCE:
                  SimpleDate              (1..1)
                  HistoricalDate          (0..1)
            ENDSEQUENCE
            SEQUENCE:
                  StartDate               (1..1)
                  HistoricalStartDate     (0..1)
                  EndDate                 (0..1)
                  HistoricalEndDate       (0..1)
            ENDSEQUENCE
            SEQUENCE:
                  EndDate                 (1..1)
                  HistoricalEndDate       (0..1)
            ENDSEQUENCE

HistoricalDateType
      NonISODate                          (1..1)
      HistoricalDateFormat                (0..1)
      Calendar                            (0..1)
```

Eight elements are defined as DateType. Three additional elements use DateType as an extension base, adding specific additional elements or attributes to define the use of the date in that particular location. Five attributes use BaseDateType. Usage locations are listed below.

Element using DateType	
datacollection.xsd	DataCollectionDate
reusable.xsd	EffectivePeriod
reusable.xsd	GeographicTime
reusable.xsd	PublicationDate
Element using extension base DateType	
datacollection.xsd	DataCollectionFrequency
reusable.xsd	AccessRestrictionDate
reusable.xsd	ReferenceDate
Attribute using BaseDateType	
reusable.xsd	authorizationDate
reusable.xsd	completionDate
reusable.xsd	validForEndDate
reusable.xsd	validForStartDate
reusable.xsd	versionDate

Date Type Examples (DateExamples.xml)
Simple Date
Complete Date Range
Range with unknown End Date
Range with unknown Start Date

4.3 - In/Out Parameters, Binding and Command Code

DDI provides a generic Command Code structure that supports the inclusion of an inline command in a specified programming language, a reference to an external file containing the command code, and the use of a StructuredCommand which serves as an extension stub for the insertion of a command code using an external namespace. CommandCode makes use of the new InParameter, OutParameter, and Binding available in primary locations throughout DDI in order to more clearly define the inputs to processes and algorithms used in collecting, altering, and revising data as it travels through the lifecycle into a data file.

The Command indicates the programming language of the command, specifies InParameters (items of information required to process the command), OutParameters, (items of information generated by the command), Binding (linking the output of an object external to the Command to the InParameter), and the CommandContent. If using a command stored as an external file, CommandFile provides the same program language information, InParameter, OutParameter, and Binding options as well as a description of the file location and a URI for the file.

Command Code

The Command indicates the programming language of the command, specifies InParameters (items of information required to process the command), OutParameters, (items of information generated by the command), Binding (linking the output of an object external to the Command to the InParameter), and the CommandContent. If using a command stored as an external file, CommandFile provides the same program language information, InParameter, OutParameter, and Binding options as well as a description of the file location and a URI for the file.

```
CommandCode
   Description                  (0..1)
   Command                      (0..1)
      ProgramLanguage           (1..1)
      InParameter               (0..n)
      OutParameter              (0..n)
      Binding                   (0..n)
      CommandContent            (0..1)
   CommandFile                  (0..1)
      ProgramLanguage           (1..1)
      InParameter               (0..n)
      OutParameter              (0..n)
      Binding                   (0..n)
      Location                  (0..1)
      URI                       (0..1)
   StructuredCommand            (0..1)
```

InParameter, OutParameter, Binding

InParameter and OutParameter share the contents of Parameter. InParameter adds the attribute limitedArrayIndex used to specify the object(s) in the zero-based array used by the Parameter. This is used only when the attribute isArray is set to "true". The Parameter provides the standard Parameter Name and Description elements. An optional Alias may be defined in cases where a command is written using a Alias. This allows a clear link between, for example, the term AGE in the command "If AGE >= 21" and in InParameter with the Alias value of "AGE". This is particularly useful when using external CommandFile scripts which would not contain the ID of the InParameter in the command. ValueRepresentation allows for a definition of the expected contents and is recommended if you are sharing data with others. DefaultValue provides a value to use if the OutParameter bound to the InParameter provides no value. Binding is a simple linkage indicating the contents of a specified SourceParameter provide the value for the TargetParameter.

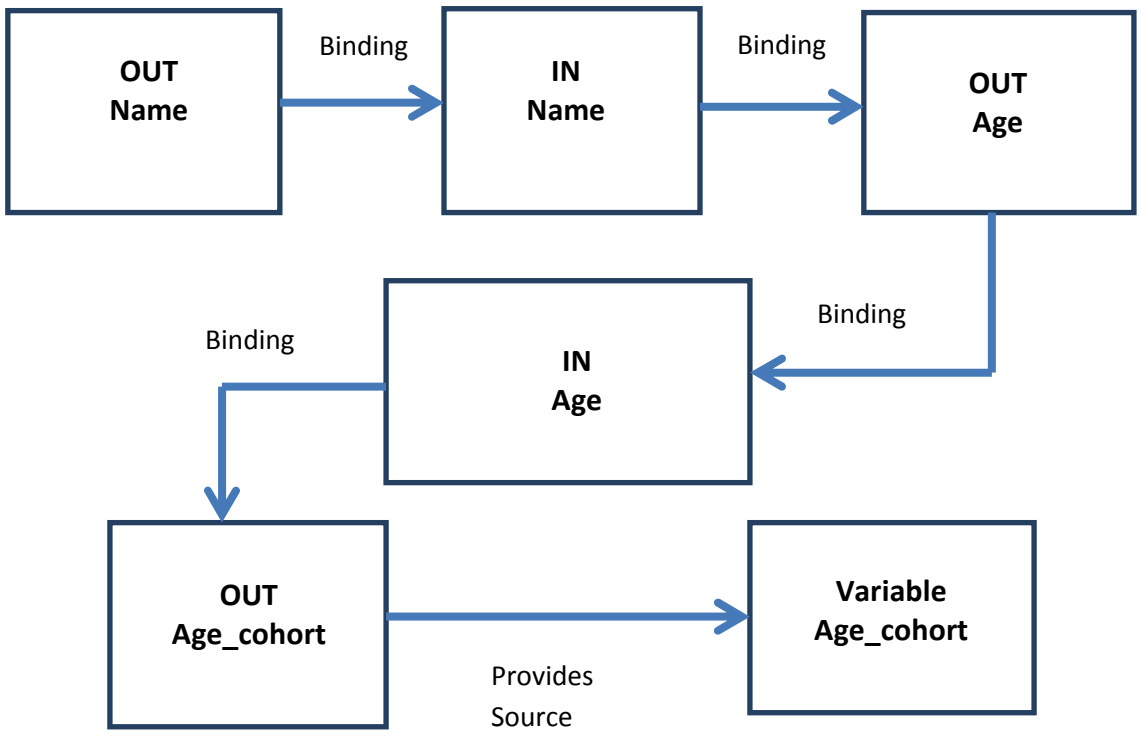
```
Parameter
   Extension base: IdentifiableType
   ParameterName                  (0..n)
   Alias                          (0..1)
   Description                    (0..1)
   ValueRepresentation            (0..1)
   DefaultValue                   (0..1)
   @isArray                       (default="false")

InParameter
   Extension base: Parameter
   @limitArrayIndex

OutParameter
   Type="Parameter"

Binding
   SourceParameterReference       (1..1)
   TargetParamterReference        (1..1)
```

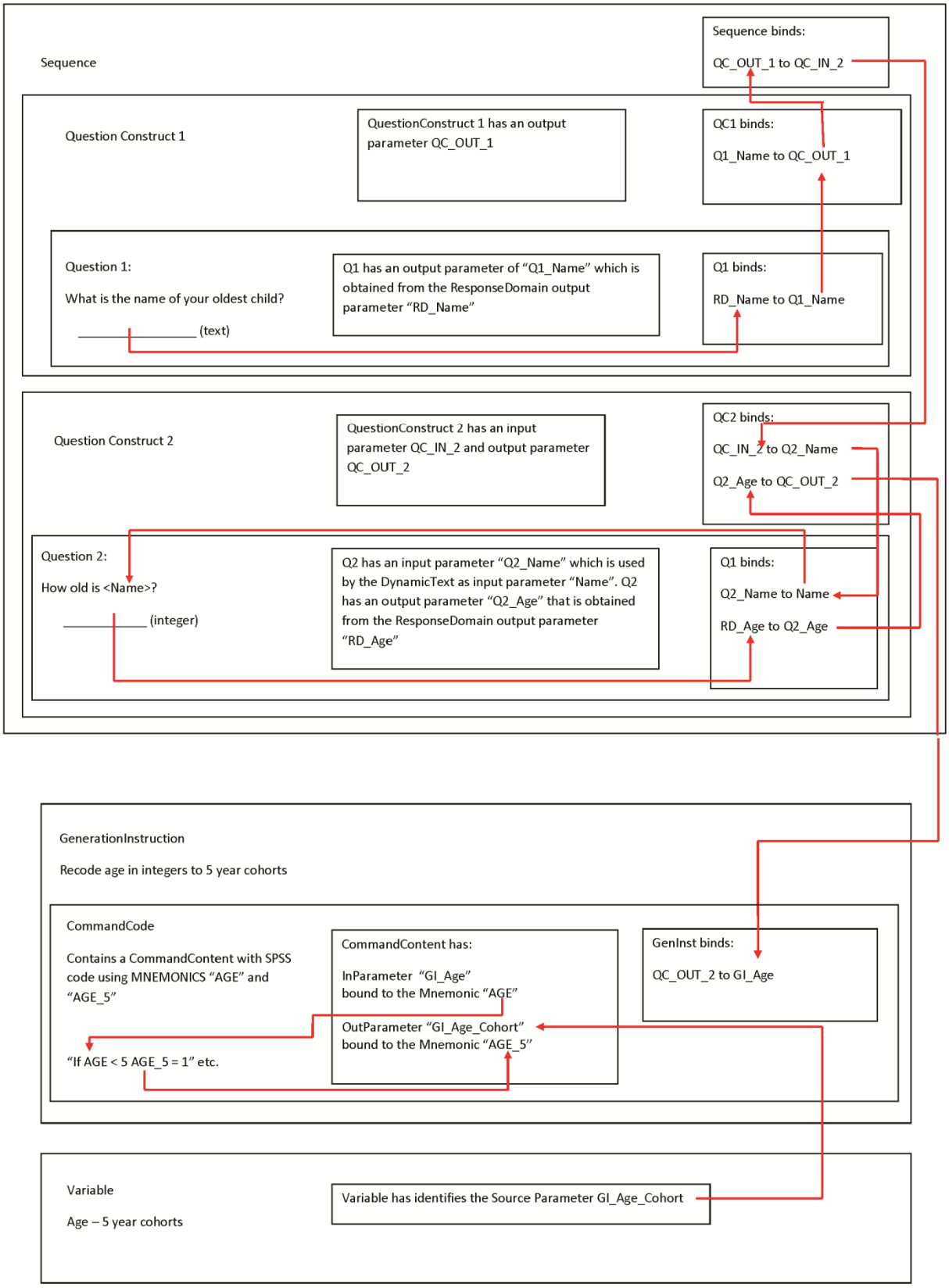
The following chart lists where each of these objects is available within DDI.

OBJECT	IN	OUT	BIND	Source Parameter Reference	Command Code
QuestionItem	x	x	x		
QuestionGrid	x	x	x		
QuestionBlock	x	x	x		
ControlConstruct	x	x	x		
IfCondition					x
UntilCondition					x
WhileCondition					x

InitialValue					x
LoopWhile					x
StepValue					x
ComputationItem					x
ControlConstructReference			x		
InterviewerInstruction	x		x		
GeneralInstruction					x
GenerationInstruction					x
Category/Generation					x
DynamicText	x				
DynamicText/Expression	x				
ResponseDomain		x			
CodeSubsetInfo					x
CommandCode	x	x	x		
Command	x	x	x		
CommandFile	x	x	x		
Variable				x	

The following example tracks the capture of a child's Name to its use in the dynamic text of a subsequent question regarding the child's age, the use of the age response by a generation instruction recoding the provided age into an age cohort, and finally the storage of the age cohort value in a variable.




Sequence
Sequence binds:
QC_OUT_1 to QC_IN_2
Question Construct 1
QuestionConstruct 1 has an output parameter QC_OUT_1
QC1 binds:
Q1_Name to QC_OUT_1
Question 1:
What is the name of your oldest child?
_____ (text)
Q1 has an output parameter of "Q1_Name" which is obtained from the ResponseDomain output parameter "RD_Name"
Q1 binds:
RD_Name to Q1_Name
Question Construct 2
QuestionConstruct 2 has an input parameter QC_IN_2 and output parameter QC_OUT_2
QC2 binds:
QC_IN_2 to Q2_Name
Q2_Age to QC_OUT_2
Question 2:
How old is <Name>?
_____ (integer)
Q2 has an input parameter "Q2_Name" which is used by the DynamicText as input parameter "Name". Q2 has an output parameter "Q2_Age" that is obtained from the ResponseDomain output parameter "RD_Age"
Q1 binds:
Q2_Name to Name
RD_Age to Q2_Age
GenerationInstruction
Recode age in integers to 5 year cohorts
CommandCode
Contains a CommandContent with SPSS code using MNEMONICS "AGE" and "AGE_5"
"If AGE < 5 AGE_5 = 1" etc.
CommandContent has:
InParameter "GI_Age"
bound to the Mnemonic "AGE"
OutParameter "GI_Age_Cohort"
bound to the Mnemonic "AGE_5"
GenInst binds:
QC_OUT_2 to GI_Age
Variable
Age – 5 year cohorts
Variable has identifies the Source Parameter GI_Age_Cohort

In/Out Parameter and Binding Example (InOutBindingExample.xml)

4.4 - Note

A Note is designed to capture local information related to one or more identifiable objects. Note is designed to be an inherent part of the DDI. The information may be passed along with a DDI object over its lifetime. It is not intended to replace formal extension mechanisms. Formal extension should be used whenever a locally used object needs to be added to the schema. However, run-time extension needs may arise and Note can be used to capture the required metadata and associate it with the appropriate object. Note may be used to capture information during processing which is then removed prior to publication. Removing a Note removes all the links to related objects within the DDI structure.

DDI recommends placing the Note within the maintainable object containing the objects referenced by the Note. Each note may indicate who is responsible for the note, its type using a controlled vocabulary, the subject of the note, a head and note content, a set of key/value pairs and language specification for the overall note. In addition each note must be related to one or more identifiable objects.

```
Note
      TypeOfNote                          (0..1)
      NoteSubject                         (0..1)
      Relationship                        (1..n)
            RelatedToReference            (1..1)
            RelationshipDescription       (0..1)
      Responsibility                      (0..n)
      Header                              (0..1)
      NoteContent                         (0..1)
      ProprietaryInfo                     (0..1)
            AttributeKey                  (1..1)
            AttributeValue                (1..1)
      @xml:lang                           (optional)
```

A Note may be classified by its type and subject. TypeOfNote should express the purpose of the note (e.g. processing note, request for review, run time extension, etc.). NoteSubject is used to express the topical subject of the Note. Both TypeOfNote and NoteSubject may be expressed using a controlled vocabulary. Relationship provides a reference to any object with an ID, preferably within the same maintainable, and a description of the relationship if needed for clarity. This is expressed as a structured string. The direction of the link between a Note and another object is always from the Note to the object. The Header provides a brief label or heading for the NoteContent which is expressed as a structured string. The provision of ProprietaryInfo (a StandardKeyValuePair) allows for the expression of the content as a tuple (attribute key and value pair) to support computer systems and applications. The overall language of the Note can be provided with the xml:lang attribute.

The following examples contain common uses of Note. The first contains a production process note which will be removed once it is acted upon. The second contains a work-around for a bug that has been reported and resolved, but not yet published as a sub-minor release. Once the correction is made to the schema, the organization will transfer the content to the corrected structure and remove the note. The

third example shows the use of ProprietaryInfo in capturing a run-time local extension. This may later be transferred to a formal extension in a local namespace at a later date and the Note removed.

Note Examples (NoteExample.xml)
Example 1: Production Process
Example 2: Work-around
Example 3: Proprietary

4.5 - Quality Descriptions

DDI addresses descriptions of quality at three levels, the quality of the metadata as captured, means of ensuring policy for processing, and the quality of the data. Statements regarding the quality standards used for processing or measuring the quality of the data related to the metadata may be referenced from the following areas: study unit, group, sub-group, resource package, methodology, various descriptions of data processing, data file creation or copying, and archival processing.

Metadata Quality

Metadata quality is captured at the level of the Maintainable object and refers to the metadata contained in that object. It is held in the complex element MetadataQuality which is found in AbstractMaintainable. It is a relatively simple statement of quality such as completeness, transcription of content from the original source, level of review or verification, etc.

```
MetadataQuality
      TypeOfMetadataQuality                  (1..1)
      MeasurePurpose                         (0..1)
      MeasureValue                           (0..1)
      Description                            (0..1)
```

Both TypeOfMetadataQuality and MeasureValue are CodeValueType and support the use of an external controlled vocabulary. These are the two primary pieces of information needed to track metadata quality internally within an organization. TypeOfMetadataQuality identifies the measure being tracked and MeasureValue provides a specific value for that measure. For example, an organization may have a number of measures one of which rates its transcription quality. It may have various values for this including: 1) Tagging and transfer of a digital content, 2) double entry, 3) single entry direct transcription, 4) selected transcription, and 5) summarization. The purpose for capturing the measure is held in MeasurePurpose (StructuredStringType) and other descriptive information that is useful to the organization or in particular a user, is held in Description (StructuredStringType).

EXAMPLE:

```
<r:MetadataQuality>
      <r:TypeOfMetadataQuality codeListName="MPCQualityTracking">
            Transcription
      </r:TypeOfMetadataQuality>
      <r:MeasurePurpose>
            <r:Content>To identify the transcription method used for
      the metadata in this section</r:Content>
```

```
      </r:MeasurePurpose>
      <r:MeasureValue
codeListName="MPCTranscriptionType">1</r:MeasureValue>
      <r:Description>
            <r:Content>Content was original digital text. Content was
tagged and processed through an XSLT to transform it to DDI
structure.</r:Content>
      </r:Description>
</r:MetadataQuality>
```

Quality Statement

Quality statements are compiled in a QualityStatementScheme which may be published within a StudyUnit, Group, or ResourcePackage. Quality statements primarily address processes and steps that are taken to ensure quality within those processes. A QualityStatement allows for either the identification of an external standard plus a statement regarding compliance with that standard, or a general statement of steps taken to ensure quality for a given process or activity. A QualityStatement is attached to a processing area by reference from the description of the process/activity itself.

```
QualityStatement
      Extension base: Versionable
      CHOICE:                                 (0..n)
            StandardsCompliance
                  Standard                    (1..1)
                  ComplianceDescription       (0..1)
            OtherQualityStatement
      ENDCHOICE
```

StandardsCompliance consists of a reference to the Standard using the structure of OtherMaterial. This could reference a document, web site, or other source containing a formal standard for processing, best practice, internal protocol, or other statement of quality. ComplianceDescription (StructuredStringType) provides details on how this standard or protocol was applied, in particular noting any deviations or issues that would have an impact on the quality factors being assessed. When no formal standard or protocol exists, use OtherQualityStatement (StructuredStringType) to describe steps taken to ensure quality. Quality statements can be referenced from the following locations and should relate quality assessment information focused on the process, activity or general coverage area where the reference is included:

a:ArchiveSpecific/r:QualityStatementReference	In this location the QualityStatement being referenced should relate to the monitoring of the Archive's activities and operations. For example: TRAC certification using the reference to the TRAC standard and noting the Archive's certification dates in the compliance section
d:Methodology/r:QualityStatementReference	In this location the QualityStatement being

	referenced should relate to overall methodology, i.e., overall scientific method followed, statistical standards,
d:CollectionEvent/r:QualityStatementReference	In this location the QualityStatement being referenced should relate to a specific collection event, i.e., respondent confidentiality protocol, use of human subjects protocol
d:ProcessingEvent/r:QualityStatementReference	In this location the QualityStatement being referenced should relate to the specific processing event, i.e., best practices for data cleaning in a specific area
g:ResourcePackage/r:QualityStatementReference	In this location the QualityStatement being referenced should relate to the specific activities surrounding the creation of the resource package, i.e., criteria for selecting the schemes to include, review processes, modification, etc.
g:ResourcePackage/r:QualityStatementScheme	The inline content of a publication package not related to a specific study (payload)
g:ResourcePackage /r:QualityStatementSchemeReference	The referenced content of a publication package not related to a specific study (payload)
g:Group/r:QualityStatementReference	In this location the QualityStatement being referenced should relate to the specific activities surrounding the creation of the Group, i.e., grouping, selection, or organizational policies.
g:Group/r:QualityStatementScheme	The scheme of QualityStatements shared by the Group included in-line
g:Group/r:QualityStatementSchemeReference	The scheme of QualityStatements shared by the Group included by reference
g:SubGroup/r:QualityStatementReference	In this location the QualityStatement being referenced should relate to the specific activities surrounding the creation of the SubGroup, i.e., grouping, selection, or organizational policies. (Allowing for the later subordination of a Group thereby turning it into a SubGroup)

g:SubGroup/r:QualityStatementSchemeReference	The scheme of QualityStatements shared by the sub-group included by reference
pi:PhysicalInstance/r:QualityStatementReference	In this location the QualityStatement being referenced should relate to the data file, i.e., a statement regarding data lost during rescue from an old media, the use of standards protocols in verification that this is a valid copy of another file
s:StudyUnit/r:QualityStatementReference	In this location the QualityStatement being referenced should relate to the specific activities surrounding the creation of the StudyUnit, i.e., coverage, purpose, organization, etc.
s:StudyUnit/r:QualityStatementScheme	The scheme of QualityStatements shared by the StudyUnit included in-line
s:StudyUnit/r:QualityStatementSchemeReference	The scheme of QualityStatements shared by the StudyUnit included by reference

Data Quality

The quality of the data collected is addressed in the ProcessingEvent containing DataAppraisalInformation. This contains two common measures of data quality from survey methodology, ResponseRate and SamplingError. It also allows for the description of other quality measures in OtherAppraisalProcess. Note that ProcessingEvent also contains a QualityStatementReference. This should be used to relay information on process quality. DataAppraisalInformation is used primarily to capture the results of data appraisal measures.

```
DataAppraisal
      ResponseRate                        (0..n)
            SampleSize                    (0..1)
            NumberOfResponses             (0..1)
            SpecificResponseRate          (0..1)
            r:Description                 (0..1)
      SamplingError                       (0..n)
      OtherAppraisalProcess               (0..n)
```

The response rate can be repeated to express differing response rates by mode of deliver, location, etc. The individual response rates may be expressed as a total sample size with number of responses and/or as a specific response rate. The description should be used to differentiate when multiple response rates are provided. SamplingError (StructuredStringType) is intended to contain a discussion of the sampling error. It may be structured to differentiate between the statement of the error itself, how it was

calculated, etc. OtherAppraisalProcess allows for the description of other measures of data appraisal as needed.

Quality Statements (QualityStatementExamples.xml)

4.43 - Questions – Item, Grid, and Block

As the complexity of question structures increase, three general question structures have been identified within DDI and are addressed by the following structures: QuestionItem, QuestionGrid, and QuestionBlock. Note that all three structures reflect the reusable structure of various question types and do not relay any information on the applied use of the question within an instrument. The applied information, including question order, is captured by the ControlConstruct structures used by the Instrument. All question structures are maintained within a QuestionScheme and can be further organized for administrative purposes by QuestionGroup (a standard grouping structure available in most schemes). Only QuestionItem, QuestionGrid, and QuestionBlock may be referenced by a QuestionConstruct. A Sequence (type of ControlConstruct) is used to order questions within an instrument and is maintained separately as a reusable object. All question structures are versionable and contain a name for the question. All response domains may be described inline while a set of commonly used structures may be captured as managed representations and be reused by reference. These include Missing Values, Text, Numeric, DateTime, and Scale (see Section 4.51 for details on all value representations and response domains).

QuestionItem	
Namespace: **d** *Parent Maintainable:* **QuestionScheme**	**Versionable**
Required Referenced Objects:	*Optionally Referenced Objects:* Category Code Concept Delineation InterviewerInstruction ManagedMissingValuesRepresentation ManagedTextRepresentation ManagedNumericRepresentation ManagedDateTimeRepresentation ManagedScaleRepresentation
QuestionItem Extension base: Versionable QuestionItemName (0..n) r:InParameter (0..n) r:OutParamter (0..n) r:Binding (0..n) QuestionText (0..n) QuestionIntent (0..1)	

CHOICE	(0..1)
CHOICE	
ResponseDomain	
ResponsedomainReference	
ENDCHOICE	
StructuredMixedResponseDomain	
CHOICE	(2..n)
ResponseText	
ResponseDomainInMixed	
CHOICE	(1..1)
ResponseDomain	
ResponseDomainReference	
ENDCHOICE	
AttachementLocation	(0..1)
CHOICE	(1..n)
r:CodeReference	
r:CategoryReference	
DomainSpecificValue	
r:Value	(1..n)
@attachementDomain	(optional)
ENDCHOICE	
@attachementBase	(optional)
ENDCHOICE	
ENDCHOICE	
ResponseCardinality	(0..1)
@minimumResponses	
@maximumResponses	
r:ConceptReference	(0..n)
ExternalAid	(0..n)
CHOICE	(0..n)
ExternalInterviewerInstruction	
InterviewerInstructionReference	
ENDCHOICE	
@estimatedMinutesResponseTime	(optional)

A QuestionItem is a basic question containing a QuestionIntent, the concept being measured by the question, text for the question, response domain information, clarifying instructions, external aids (clarifying objects used in presenting the question to the respondent), Input and Output Parameters and Bindings, allowed response cardinality and an estimation of the time required to respond. The QuestionText is a DynamicTextString. Note that due to the variation of content in the QuestionText that is required to illicit equivalent responses in different languages, QuestionText is repeatable. A StructuredMixedResponseDomain supports situations where a response may include more than one type of response domain. This could be the inclusion of a set of Missing Values (responses that capture non-response and other values to be treated as invalid during analysis), or the commonly used addition

of a text response to capture specific information when the code or category response is “Other”. StructuredMixedResponse allows for the addition of text and/or another response domain to be attached to one or more values of the primary response domain as well as stacking combinations of response options with intervening text.

QuestionItem Examples:

1. A generic simple QuestionItem managed within a QuestionScheme (link)
2. A question with a StructuredMixedResponseDomain (link)

QuestionGrid	
Namespace: **d** *Parent Maintainable:* **QuestionScheme**	**Versionable**
Required Referenced Objects:	*Optionally Referenced Objects:* Category Code Concept Delineation InterviewerInstruction ManagedMissingValuesRepresentation ManagedTextRepresentation ManagedNumericRepresentation ManagedDateTimeRepresentation ManagedScaleRepresentation

QuestionGrid	
Extension base: Versionable	
QuestionGridName	(0..n)
r:InParameter	(0..n)
r:OutParamter	(0..n)
r:Binding	(0..n)
QuestionText	(0..n)
QuestionIntent	(0..1)
GridDimension	(0..n)
CHOICE	(1..1)
CodeDomain	
Roster	
r:Label	(0..1)
ConditionForContinuation	(0..1)
@baseCodeValue	(required)
@codeInterationValue	(required)
@minimumRequired	default=”1”
@maximumAllowed	default=”1”
ENDCHOICE	
@rank	(required)
@displayCode	default=”true”
@displayLabel	default=”true”
CHOICE	(0..1)

```
        CHOICE
              ResponseDomain
              ResponsedomainReference
        ENDCHOICE
        StructuredMixedGridResponseDomain
              CHOICE                                              (0..n)
                    GridResponseDomain
                          CHOICE                                  (1..1)
                                ResponseDomain
                                ResponseDomainReference
                          ENDCHOICE
                          GridAttachement                         (0..n)
                                CHOICE                            (0..n)
                                      SpecificCellCoordinate
                                      CellCoordinatesAsDefined
                                            SelectDimension       (0..n)
                                                  @rank           (required)
                                                  @allValues
                                                  @specificValue
                                                  @rangeMinimum
                                                  @rangeMaximum
                                ENDCHOICE
                                @allCells                         default=”false”
                    NoDataByDefinition
              ENDCHOICE
  ENDCHOICE
  CellLabel                                                       (0..n)
        Extension base: r:LabelType
        Attachement                                               (0..n)
              CHOICE                                              (0..n)
                    SpecificCellCoordinate
                    CellCoordinatesAsDefined
                          SelectDimension                         (0..n)
                                @rank                             (required)
                                @allValues
                                @specificValue
                                @rangeMinimum
                                @rangeMaximum
              ENDCHOICE
              @allCells                                           default=”false”
  r:ConceptReference                                              (0..n)
  ExternalAid                                                     (0..n)
  CHOICE                                                          (0..n)
        ExternalInterviewerInstruction
        InterviewerInstructionReference
  ENDCHOICE
  @estimatedMinutesResponseTime                                   (optional)
```

A QuestionGrid provides a multidimensional structure used to capture a complex response. In many cases these are simple structures, for example a question regarding an assessment of each of a list of candidates for political office. However, some grids are complex and capture different responses for each of a list of items. In addition, the list may be provided by the question or by the respondent (indicated by the use of a "roster" of blank text items). A QuestionGrid contains the basic elements of the QuestionItem but rather than a ResponseDomain or StructuredMixedResponseDomain contains a GridDimension that sorts one or more ResponseDomains or StructuredMixedGridResponseDomains into a specific dimension of the grid. The structure is similar to the NCube dimensional structure using code domain or roster rather than variables to structure the dimension, and then providing a response domain for each cell in the grid. A single ResponseDomain indicates that this is the response domain for each cell in the grid. Note that an internal cell label may be added to any cell for clarification purposes.

QuestionGrid Examples:

1. A simple QuestionGrid capturing a scale response for each of 3 candidates (link)
2. A QuestionGrid using a roster to capture the categories for which responses are gathered (link)
3. A complex QuestionGrid with cell labels and secondary responses for specific responses (link)

QuestionBlock	
Namespace: **d** *Parent Maintainable:* **QuestionScheme**	**Versionable**
Required Referenced Objects:	*Optionally Referenced Objects:* Concept Delineation InterviewerInstruction QuestionItem QuestionGrid
QuestionBlock	
Extension base: Versionable	
QuestionBlockName	(0..n)
r:InParameter	(0..n)
r:OutParamter	(0..n)
r:Binding	(0..n)
QuestionBlockIntent	(0..1)
CHOICE	(0..n)
StimulusMaterial	
QuestionItemReference	
QuestionGridReference	
ENDCHOICE	
QuestionSequence	(0..n)
ResponseCardinality	(0..1)
@minimumResponses	
@MaximumResponses	
r:ConceptReference	(0..n)

ExternalAid	(0..n)
CHOICE	(0..n)
ExternalInterviewerInstruction	
InterviewerInstructionReference	
ENDCHOICE	
@estimatedMinutesResponseTime	(optional)

A QuestionBlock is intended to bundle together a set of questions (items and/or grids) that have meaning only in relation to a specified object expressed as the stimulus material. This form of question set is common in educational testing where a text, image, or other material is provided and the respondent is asked questions specific to the material. For example, a portion of a play script is provided and the respondent is asked question concerning the dialog and/or stage directions provided in the script. Note that the intent of QuestionBlock is NOT to bundle together a set of questions that are commonly used together or used in a specified order. A Sequence (type of control construct) is the appropriate way to manage this type of set relationship. It is assumed that one or more StimulusMaterial objects will be used. The 0..n cardinality of the CHOICE of StimulusMaterial, QuestionItemReference, and QuestionGridReference is provided simply to support the use of DDI during the development process when this piece of information may be unavailable. StimulusMaterial is part of this choice to allow for its insertion at any location before, after, or in the midst of the referenced Questions.

QuestionBlock contains the standard elements of QuestionItem and QuestionGrid and then structures any combination of StimulusMaterial with referenced QuestionItems and QuestionGrids. It allows for an indication of QuestionSequence (i.e., a means of declaring the sequence is important or that it can be randomized or otherwise altered).

QuestionBlock Examples:

1. A QuestionBlock organizing QuestionItems and QuestionGrids managed within a QuestionScheme. The example reflects the contents of a question block from PISA (educational test: TAKE THE TEST: SAMPLE QUESTIONS FROM OECD'S PISA ASSESSMENTS - ISBN 978-92-64-05080-8 - © OECD 2009) depicting an extract from a script, a list of technical terms, and a drawing of a stage set along with questions related to the activities and positions of actors during the play. The related category, code and delineation information is provided for clarity. (link)

Question Examples (QuestionExample.xml)

4.6 - Represented Variable

A scheme of reusable Represented Variables was added to Logical Product in order to better support ISO/IEC 11179 by the maintenance and reuse of the basic components of variables. It is anlages to the GSIM Represented Variable. A Represented Variable consists of a reference to a Conceptual Variable or

to a Concept and Universe (the components of a Conceptual Variable) plus a definition of its representation expressed as a Category Scheme or a Value Representation. This limited structure makes the DDI Represented Variable more generic than the full ISO/IEC 11179 model as it does not include a Represented Variable example, derivation, or derivation rule. Enumerated values are expressed by reference to a Category Scheme or by use of a Code Representation (which contains references to the associated categories). Non-enumerated values may be described broadly or as specifically as supported by the Value Representation structure.

Represented Variables are managed within a Represented Variable Scheme containing all of the common Scheme structures including a Represented Variable Group which allows for associating a group of Represented Variables with a specific Concept or Universe and/or a set of Subjects and Keywords. As with other Groups of objects within Schemes, the Represented Variable Group can be typed and represent order and un-ordered lists or hierarchies.

```
RepresentedVariable
      Extension base: VersionableType
      RepresentedVariableName                (0..n)
      r:Label                                (0..n)
      r:Description                          (0..1)
      r:UniverseReference                    (0..1)
      CHOICE
         r:ConceptualVariableReference       (0..1)
         SEQENCE
            r:UniverseReference              (0..1)
            r:ConceptReference               (0..1)
         ENDSEQUENCE
      ENDCHOICE
      CHOICE                                 (0..1)
         r:CategorySchemeReference
         r:ValueRepresentation
         r:ValueRepresentationReference
      ENDCHOICE
```

While a Represented Variable by definition requires a Universe (Object), Concept, and Representation (Value Domain) these are optional in DDI to support production work during which all references may not be available. Note that when referenced by a Variable, the Variable may further constrain the Concept, Universe, and Value Representation. For example: The Concept of Age to a subordinate concept of Physiological Age; Universe of Persons to Persons residing within Luxembourg; and Numeric Representation of single year increments to top coding of 99.

Represented Variable – with usage by Variable (RepresentedVariable.xml)

Represented Variable Example 1 uses references to a Concept, Universe, and Category Scheme. Note that the if there is a Conceptual Variable containing links to the same Concept and Universe that these two references in the Represented Variable may be replaced by a reference to the Conceptual Variable. The DDI Represented Variable is built using a set of references to a previously defined Universe, Concept and optional Category Scheme. The Universe references should reflect the broadest universe applicable,

i.e., Persons. If the Concept is represented by a set of categories expressed by a CategoryScheme the scheme may be referenced. Represented Variable Example 2 provides a description of the representation type, in this case numeric.

The Represented Variable is a conceptual model of the combination of a concept linked to a specific universe or object that is represented in a particular way. A Variable is the applied use of a Represented Variable where the universe may be further refined (Variable Example 1) and the exact parameters of the representation are defined in greater detail (Variable Example 2). The DDI Represented Variable reflects the structure of the ISO/IEC 11179 Data Element. Note that the Value Representation of the Represented Variable may be broadly defined. The exact details and/or content restrictions for the representation may be added when referenced by a Variable used within a specific application. The example below shows a Represented Variable and two Variables which reference it within different contexts.

4.7 - Statistical Summary

StatisticalSummary provides a set of summary statistics calculated for the variables in a data set and is located in PhysicalInstance as it reflects the contents of a specific data file. The summary statistics may be provided in one of three ways; in-line using the VariableStatistics, by referencing a related PhysicalInstance containing the VaraibleStatistics for the same data file content, or by referencing a related PhysicalInstance for a data file that contains the summary statistics as a separate object set. The second two options use the structure StatisticalDataLocation which is a reference to the PhysicalInstance plus a Boolean attribute “isInline”. If set to “true” this indicates that the summary statistics are contained in the data file associated with the referenced PhysicalInstance.

VariableStatistics supports the following information for any variable: the total responses, a choice of weights applied when calculating the summary statistics, identification of missing values, the summary statistics for the variable, and category statistics with an optional filter. Category statistics may be filtered by a single variable (i.e., filtering the category statistics in a multinational data file by country) using FilteredCategoryStatistics. Expressing statistics that require the use of multiple filters should be handled as a cross-tabulation and captured in an NCube structure.

Note that while a number of statistical summaries may be included within a single VariableStatistics object, you cannot use more than a single weight (either a standard weight or a weight variable). However you are able to provide both weighted and unweighted values for each statistic. If a complex weighting structure is used based on multiple weights, a virtual variable should be created that expresses the combined use of the weights involved. The virtual variable would reference a generation instruction containing the process.

```
StatisticalSummaryType
      StatisticalDataLocation                                  (0..n)
            r:PhysicalInstanceReference                        (1..1)
            @isInline
      VariableStatitics                                        (0..n)
            Extension base: Identifiable
```

```
VariableReference                                    (1..1)
TotalResponses                                       (0..1)
CHOICE                                               (0..1)
      StandardWeightReference
      WeightVaraibleReference
ENDCHOICE
MissingValuesReference                               (0..1)
SummaryStatistic                                     (0..n)
      TypeOfStatistic                                (1..1)
      Statistic                                      (0..1)
            @isWeighted
            @computationBase
UnfilteredCategoryStatistics                         (0..n)
      VariableCategory                               (0..n)
            CategoryValue                            (1..1)
            CategoryStatistic                        (0..n)
                  TypeOfCategoryStatistic            (1..1)
                  Statistic                          (0..1)
                        @isWeighted
                        @computationBase
FilteredCategoryStatistics                           (0..n)
      FilterVariableReference                        (0..1)
      FilterVariableCategory                         (0..n)
            FilterCategoryValue                      (1..1)
            VariableCategory                         (0..n)
                  CategoryValue                      (1..1)
                  CategoryStatistic                  (0..n)
                        TypeOfCategoryStatistic      (1..1)
                        Statistic                    (0..1)
                              @isWeighted
                              @computationBase
```

The example is for the following summary and category statistics. The general contents of the referenced items are provided.

Summary Statistics					**Standard Weight**
V1	count = 100				ID=SW
	weighted count = 1000				Value = 10
Unfiltered Category Statistics					**Variable Gender**
	wtCount	weighted %			ID=V1
Male	450	0.45			1=Male
Female	550	0.55			2=Female
V2	count = 100				
	weighted count = 1000				
Unfiltered Category Statistics					**Variable Region**
	wtCount	weighted %			ID=V2

North	500	0.5			a=North
South	500	0.5			b=South
Gender (V1) Filtered by Region (V2)					
	North		South		
	count	col %	count	col %	
Male	20	0.4	25	0.5	
Female	30	0.6	25	0.5	

Statistical Summary Example (StatisticsExample.xml)

4.8 – Variable Value Representation and Question Response Domain

Representations describe the structure and content of data as it is captured from the population and held within a data file. They share common category schemes and code list as well as the means of defining numeric ranges and text content. DDI begins by defining the core descriptive content for a wide range of representations and then adds a set of common objects used by all Value Representations (Variables) or Response Domains (Questions) in their applied use. A number of these representations also have a Managed version which allows a single description of a Representation to be reused within and between studies. Some Representations reference reusable structures (Category Schemes, Code Lists, etc.) and are already reusable. Others such as Numeric Representations have a Managed Representation contained in a ManagedRepresentationScheme located in Logical Product. This Scheme contains all forms of Managed Representations and supports the standard Scheme organizational objects that allow for grouping and classification according to Subjects, Keywords, Concept, and Universe.

This section describes the types of Representation available for use along with the added content for use as a Value Representation or Response Domain. This is followed by a full description each Representation Base type and a description of Managed Representation types. Note that wherever the substitution base ValueRepresentation or ResponseDomain is used there is also an option for a ValueRepresentationReference or ResponseDomainReference. Only Representations containing managed content (ManagedRepresentation types, CategoryScheme, CodeList, GeographicStructureCode, and GeographicLocationCode) can be mapped for comparison purposes. This excludes most Representation types that are used solely as Response Domains.

ValueRepresentation and ResponseDomain are the applied uses of Representations.

ValueRepresentation (abstract)

ValueRepresentation serves as the abstract head of a substitution group ValueRepresentation. Any member of the substitution group can be used as a substitution for r:ValueRepresentation wherever it occurs. All members of this group use r:RepresentationType as their extension base thereby providing a standard set of objects for each ValueRepresentation.

```
RepresentationType
  RecommendedDataType   (0..1)
  GenericOutputFormat   (0..1)
  ContentDateOffset     (0..1)
  @missingValue         optional
  @blankIsMissingValue  optional
  @classificationLevel  optional
                        [Nominal|Ordinal|Interval|Ratio|Continuous]
```

All ValueRepresentations contain a base set of objects describing its applied use. The RecommendedDataType is of type CodeValue and should contain a value reflecting a data type value (recommended: W3C XML Schema Part 2, but excluding string sub-types, QNAME, and NOTATION). The actual data type of the stored data content may vary (for example be of a broader type), but the purpose of this element is to capture the data type intended by the originator of the data. Likewise GenericOutputFormat allows the originator to provide guidance regarding the displayed format of the variable content. ContentDateOffset provides an alternate referent date for the variable content. For example, in a population survey the data may generally be collected for a particular date but some items such as the response to "Where did you live 5 years ago?" refers to a negative offset of 5 years from the general referent date. The attribute classificationLevel allows for definition of the variable content as Nominal, Ordinal, Interval, Ratio, or Continuous in nature.

Note that the two attributes @missingValue and @blankIsMissingValue have been retained from DDI 3.1 to support users for whom the shift to the new separation of missing (invalid) values from valid values would be problematic. Best practice strongly encourages the use of the separate MissingValueRepresentation to differentiate valid from invalid values.

ValueRepresentation substitutions

All value representation substitutions contain the basic ValueRepresentation objects plus a reference to one or more of the specific delineation of the same type. Note that if multiple representations are referenced they must not have duplicated values. These include:

- CodeRepresentation
- DateTimeRepresentation
- GeographicLocationRepresentation
- GeographicStructureRepresentation
- NumericRepresentation
- TextRepresentation

Within a Variable the representation of missing values is handled separately as a direct reference to a MissingValuesDelineation structure. This may also be declared as a default MissingValues within a LogicalRecord or within a PhysicalInstance.

ResponseDomain (abstract)

Response Domain serves as an abstract head of the substitution group Response Domain. Any member of the substitution group can be used as a substitution for d:ResponseDomain wherever it occurs. All members of this group use a specified Representation Base Type (ex. NumericRepresentationBaseType) as their extension base thereby providing the same set of content as contained in related Value Representation. All members of the substitution group ResponseDomain also provide the following objects.

```
r:Label                (0..n)
r:Description          (0..1)
r:OutParameter         (0..1)
r:ResponseCardinality (0..1)
    @maxResponses
    @minResponses
r:ContentDateOffset    (0..1)
```

All Response Domains can designate the intended cardinality of responses as a statement of minimum and maximum number of allowed responses. The OutParameter provides an ID for the response (or response array) so that it can be bound to the InParameter of an instruction or command (see Input/Output Parameters and Command Code for usage details). ContentDateOffset provides an alternate referent date for the question response content.

ResponseDomain substitutions

All response domain substitutions contain the basic ResponseDomain objects plus the contents of one of the specific Representation Base of the same type. Note that when using multiple domains in the StructuredMixedResponseDomain the multiple domains must not have duplicated values. Available Response Domains include:

- CategoryDomain
- CodeDomain
- DateTimeDomain
- DistributionDomain
- GeographicDomain
- GeographicStructureDomain
- GeographicLocationDomain
- LocationDomain
- MissingValuesDomain
- NominalDomain
- NumericDomain
- RankingDomain
- ScaleDomain
- TextDomain

Representation Base Types

A range of Representation Base Types are available. Each type is described as a specified Representation Base Type. All of these have related specific Response Domains, but not all have specific Value Representations. This reflects the way that the data captured by a question is represented within a data set. For example, a question may ask for a check mark to be made next to a category value which is later coded to a value which represents the category such as, M=Male. The ValueRepresentation would use the CodeRepresentation to define the valid values entered in the data file. The following list of Representation Base Types defines the usage options for each type:

Representation Base Types	**Usage options**
CodeRepresentationBaseType	ValueRepresentation or ResponseDomain
DateTimeRepresentationBaseType	ValueRepresentation or ResponseDomain
GeographicLocationCodeRepresentationBaseType	ValueRepresentation or ResponseDomain
GeographicStructureCodeRepresentationBaseType	ValueRepresentation or ResponseDomain
NumericRepresentationBaseType	ValueRepresentation or ResponseDomain
TextRepresentationBaseType	ValueRepresentation or ResponseDomain
CategoryRepresentationBaseType	ResponseDomain only
DistributionRepresentationBaseType	ResponseDomain only
GeographicRepresentationBaseType	ResponseDomain only
LocationRepresentationBaseType	ResponseDomain only
NominalRepresentationBaseType	ResponseDomain only
RankingRepresentationBaseType	ResponseDomain only
ScaleRepresentationBaseType	ResponseDomain only
MissingValuesRepresentationBaseType	MissingValueRepresentation or ResponseDomain

Note that all Representation Base Types are never used directly (i.e. there is no "CodeRepresentationBase" but a "CodeRepresentation" of type "CodeRepresentationBaseType"). All Representation Base Types have an extension base of Representation Type and therefore all contain the following objects preceding any specific content:

```
RecommendedDataType   (0..1)
GenericOutputFormat   (0..1)
ContentDateOffset     (0..1)
@missingValue         optional
@blankIsMissingValue  optional
@classificationLevel  optional
                      [Nominal|Ordinal|Interval|Ratio|Continuous]
```

CodeRepresentationBase

Defines a CodeRepresentation by referencing a CodeList and describing the valid code subset used. For example, the complete CodeList, a specified level or range, or only the most discrete codes in the list.

```
CodeRepresentation
   Extension base: RepresentationType
   CodeListReference                    (0..1)
   CodeSubsetInformation                (0..1)
      IncludedLevel                     (0..n)
      IncludedCode                      (0..1)
         CodeReference                  (0..n)
         Range                          (0..n)
            RangeUnit                   (0..1)
            MinimumValue                (0..1)
            MaximumValue                (0..1)
      DataExistence                     (0..1)
         CHOICE                         (1..1)
            LevelNumber
            DiscreteCategory            fixed="true"
         ENDCHOICE
```

References the CodeList used by the Representation and defines the portion of the CodeList used by the CodeSubsetInformation. CodeSubsetInformation allows for the specification of a level number from the CodeList to be included in the RepresentationBase, included codes defined as a range, or the specification of the just the most discrete data codes. The Range specifies the unit of the range specification as well as a minimum and maximum value. Note that these values use and extended form of Value which allows for the declaration of significant leading or trailing white space within the value as well an attribute noting if the value is inclusive (i.e., included as a valid value in the range specification). DataExistence is specified by the lowest level number for regular hierarchies or by selecting those Code items with the attribute isDiscrete="true" from the CodeList for irregular hierarchies.

See VariableScheme / CodeRepresentation in example file (link at end of section)

DateTimeRepresentation

Defines a DateTimeRepresentation by prescribing its structure and content coverage.

```
DateTimeRepresentation
   Extension base: RepresentationType
   DateTimeFieldFormat                (0..1)
   DateTimeCode                       (1..1)
   @regExp                            (optional)
```

The DateTimeFieldFormat is a CodeValue which describes the format of the date field, in formats such as YYYY/MM or MM-DD-YY, etc. If this element is omitted, then the format is assumed to be the XML Schema format corresponding to the type attribute value. The use of an external controlled vocabulary is strongly recommended. The DateTimeCode is a CodeValue and is required. This is a standard XML date type code for example date, dateTime, gYearMonth, gYear, and duration. The use of an external controlled vocabulary is strongly recommended.

See ManagedRepresentationScheme / ManagedDateTimeRepresentation in example file (link at end of section)

GeographicLocationCodeRepresentation

This Representation allows for the direct use of the contents of a Geographic Location Value as a GeographicLocationCodeRepresentation or a GeographicLocationCodeDomain. This relieves the user of creating a secondary Code List reflecting the same information and retains contextual information in the use of Geographic Locations as response domains or representations. References the GeographicLocation used, identifies which code is being used based on the AuthorizationSouce and allows specifying which codes to exclude from a set, similar to specific object exclusion from a Scheme Reference.

```
GeographicLocationCodeRepresentation
   Extension base: RepresentationType
   IncludedGeographicLocationCodes          (0..1)
      AuthorizedSourceReference             (0..1)
      GeographicLocationReference           (0..1)
      ExcludedLocationValueReference        (0..n)
   LimitedCodeSegmentCaptured               (0..n)
      Description                           (0..1)
      @arrayBase                            required [0|1]
      @startPosition                        required
      @length                               required
```

Note that the Representation references a single location type. The use of the Representation as a response domain or value representation may include the complete code or a component segment of the complete code. When used for a Response Domain if the full unique hierarchical string (i.e. State—County—Tract) is being collected as a single object then a single Representation can be used. However, if the captured data will be stored as separate variables use a StructuredMixedResponseDomain in a Question using one GeographicLocationCodeRepresentation for each segment of the complete code.

GeographicLocationCodeRepresentation provides a LimitedCodeSegmentCapture which is used to identify the segment of a Geographic Location Code which is captured in this domain. For example, a County's unique location code may be a composite of a State code (2 characters) + County code (3 characters). LimitedCodeSegmentCapture provides a description of the code segment captured in the response and specifies it through the following attributes: arrayBase (clarifying the array based used when determining the start position in the code), startPosition (the first character of the captured code), and the length (the number of characters making up the captured code). Using the above example this would be expressed as:

```
<LimitedCodeSegmentCapture arrayBase="1" startPosition="3"
length="3">
      <Description><Content xml:lang="eng">Unique code is a
      composite of a 2 character State code and 3 character
      Country code. This response domain captures ONLY the county
      code portion of the unique code</Content></Description>
</LimitedCodeSegmentCapture>
```

GeographicLocationRepresentation also provides a LimitedCodeSegmentCapture which is used to identify the segment of a Geographic Location Code which is captured in this domain. See Response Domain section above for description of its use.

See VariableScheme / GeographicLocationCodeRepresentation in example file (link at end of section)

This shows a GeographicLocationCodeRepresentation that contains the full required code for the unique identification of a county, both State and County codes. If the example also contained the LimitedCodeSegmentCapture described above the ValueRepresentation or ResponseDomain using this description would capture ONLY the 3 character County Code portion of the unique string. It would have to be paired with a State code in order to uniquely identify the County.

GeographicStructureCodeRepresentationBase

RepresentationBase for the direct use of a GeographicStructureCode as a GeographicStructureRepresentation or GeographicStructureDomain. This relieves the user of creating a secondary Code List reflecting the same information and retains contextual information in the use of Geographic Structures as response domains or representations. References the GeographicStructure used, identifies which code is being used based on the AuthorizationSouce and allows specifying which codes to exclude from a set, similar to specific object exclusion from a Scheme Reference.

```
GeographicStructureCodeRepresentationBase
   Extension base: VersionableType
   GeographicStructureCodeRepresentationBaseName  (0..n)
   Label                                          (0..n)
   Description                                    (0..1)
   IncludedGeographicStructureCodes               (0..1)
      AuthorizedSourceReference                   (0..1)
      GeographicStructureReference                (0..1)
      ExcludedGeographicLevelReference            (0..n)
```

Note that a single value representation or response domain can contain only a single code set for the structure which is identified by its Authorization Source. If a single agency manages several code types they should be clearly differentiated with separate Authorization Source identifiers (i.e., specified down to the specific coding list).

See VariableScheme / GeographicStructureCodeRepresentation in example file (link at end of section)

NumericRepresentationBase

Defines a NumericRepresentationBase by describing the valid numeric range, expressing top or bottom codes, and the valid type for the content.

```
NumericRepresentationBase
   Extension base: VersionableType
   NumericRepresentationBaseName       (0..n)
   Label                               (0..n)
   Description                         (0..1)
   NumericRange                        (0..n)
   NumericTypeCode                     (1..1)
```

Provides the valid numeric range in terms of a High and Low number, TopCode or Bottom code, as well as constraining the content through use of a controlled vocabulary. The NumericTypeCode provides definition of the W3C XML numeric type such as integer, decimal, etc.

See ManagedRepresentationScheme / ManagedNumericRepresentation in example file (link at end of section)

TextRepresentationBase

Defines a TextRepresentationBase used by a TextRepresentation or TextDomain, describing the maximum and minimum length of the text string, and providing a regular expression to further constrain the content.

```
TextRepresentationBase
   Extension base: VersionableType
   TextRepresentationBaseName          (0..n)
   Label                               (0..n)
```

```
   Description                          (0..1)
   @maxLength                           optional
   @minLength                           optional
   @regExp                              optional
```

Text allows for the definition of a minimum and maximum length of the text object as well as constraining the allowed content through use of a regular expression.

See ManagedRepresentationScheme / ManagedTextRepresentation in example file (link at end of section)

CategoryRepresentationBase

Defines a CategoryRepresentationBase by specifying the Category Scheme used.

```
CategoryRepresentationBase
   Extension base: VersionableType
   CategoryRepresentationBaseName     (0..n)
   Label                              (0..n)
   Description                        (0..1)
   CategorySchemeReference            (0..1)
```

References a CategoryScheme allowing for the exclusion of any specified object within the scheme.

See QuestionScheme / CategoryDomain in example file (link at end of section)

DistributionRepresentationBase

Defines a distribution structure used as a response domain, indicating the total amount to be distributed among the response objects.

```
DistributionRepresentationBase
   Extension base: VersionableType
   DistributionRepresentationBaseName  (0..n)
   Label                               (0..n)
   Description                         (0..1)
   DistributionValue                   (1..1)
   @decimalPositions
```

The DistributionValue provides the total value (xs:decimal) to be distributed among the response objects. The decimalPositions attribute clarifies the level of detail allowed in terms of the number of decimals accepted within a response.

See QuestionScheme / DistributionDomain in example file (link at end of section)

GeographicRepresentationBase

A specialized RepresentationBase that contains the basic information required to collect geographic information from a GIS or similar system. Provides default values as well as fields to capture case specific deviations from the default.

```
GeographicRepresentationBase
   Extension base: VersionableType
   GeographicRepresentationBaseName    (0..n)
   Label                               (0..n)
   Description                         (0..1)
   Datum                               (1..1)
   CoordinateSytem                     (1..1)
   CoordinateZone                      (1..1)
   CoordinateSource                    (1..1)
   ErrorCorrection                     (1..1)
   Offset                              (1..1)
   GeoreferenceObject                  (1..1)
   AddressMatchType                    (0..1)
   CoordinatePairs                     (1..n)
   AlternateOffset                     (0..1)
   AlternateObject                     (0..1)
   AlternateCoordinateSystem           (0..1)
   @pointFormat                        required
   @spatialPrimitive                   required
                                       (Point|Polygon|Line|LinearRing)
```

The following objects define the default values defined for the response domain: Datum identifies the geographic datum type of the object (recommend use of controlled vocabulary), CooridnateSystem identifies the coordinate system used by the response domain, CoordinateZone specifies the geographic coordinate zone used, the source of the coordinate reading is supplied in CoordinateSource, the standard offset is given in Offset, and the object used for identifying the point of the coordinate being collected is listed in the GeoreferenceObject (i.e., front door or centroid). If an address match is used AddressMatchType specifies the type of matching used. CoordinatePairs provides the capture structure for the case content. The attributes pointFormat and spatial primitive specify the format structure of the point and the spatial type being captured (Point, Polygon, Line, or Linear Ring). AltenateOffset, AlternateObject, and AlternateCoordinateSystem provide capture points for case specific information when the default values are not used.

See QuestionScheme / GeographicDomain in example file (link at end of section)

LocationRepresentationBase

Defines a mark and the region within an object (i.e., image, text, etc.) where the mark should occur. Primarily used as a response domain within a QuestionBlock.

```
LocationRepresentationBase
   Extension base: VersionableType
   LocationRepresentationBaseName      (0..n)
   Label                               (0..n)
   Description                         (0..1)
   Object                              (0..1)
   Action                              (0..n)
      RegionOfAction                   (0..1)
         Texutal                       (0..n)
         Audio                         (0..n)
         Video                         (0..n)
         XML                           (0..n)
         ImageArea                     (0..n)
      Description                      (0..1)
```

Object specifies the object upon which the action takes place. Action describes the action(s) which take place. Action specifies the region within which the action takes place described in terms of a start, stop, or region definition appropriate to each type as well as a description of the action itself.

See QuestionScheme / LocationDomain in example file (link at end of section)

NominalRepresentationBase

Defines a nominal response that is not coded or related to a particular category scheme. Used primarily by QuestionGrid, this defines a response where there is a simple check or other demarcation expressing a binary “yes | no” or “true | false” response.

```
NominalRepresentationBase
   Extension base: VersionableType
   NominalRepresentationBaseName       (0..n)
   Label                               (0..n)
   Description                         (0..1)
   @regExp                             (0..1)
```

A simple description of a nominal response which may be constrained by a regular express to a specified mark.

See QuestionScheme / NominalDomain in example file (link at end of section)

RankingRepresentationBase

Defines a ranking structure used as a response domain, indicating the ordering options for the response.

```
RankingRepresentationBase
   Extension base: VersionableType
```

```
   RankingRepresentationBaseName       (0..n)
   Label                               (0..n)
   Description                         (0..1)
   RankingRange                        (1..1)
      RangeUnit                        (0..1)
      MinimumValue                     (0..1)
      MaximumValue                     (0..1)
      @maximumRepetitionOfSingleValue  (default="1")
```

The RankingRange is an extension of Range adding the attribute maximumRepetitionOfSingleValue. The RankingRange specified the unit used for expressing the rank, provides a minimum and maximum value for the rank, and specifies how many items may have the same rank (default="1"). The Range specifies the unit of the range specification as well as a minimum and maximum value. Note that these values use and extended form of Value which allows for the declaration of significant leading or trailing white space within the value as well an attribute noting if the value is inclusive (i.e., included as a valid value in the range specification).

See QuestionScheme / RankingDomain in example file (link at end of section)

ScaleRepresentationBase

Defines a range of scale based responses varying by display, number of dimensions, and anchors.

```
ScaleRepresentationBase
   Extension base: VersionableType
   ScaleRepresentationBaseName         (0..n)
   Label                               (0..n)
   Description                         (0..1)
   ScaleDimension                      (0..1)
      Label                            (0..n)
      CHOICE                           (0..1)
         NumberRange
         Range
      ENDCHOICE
      Anchor                           (0..n)
         CategoryReference             (0..1)
         @value
      MarkedIncrement                  (0..1)
      ValueIncrement                   (0..1)
   DimensionIntersect                  (0..n)
   DisplayLayout                       (0..1)
```

Scale layouts may affect the validity and comparability of the data captured. ScaleRepresentationBase allows a specific definition of each dimension of the scale, the DimensionIntersect for multi-dimensional scales, and the specific of the scale layout. The ScaleDimension indicates the complete numeric range or textual range, the anchor for the scale expressed as a category and/or value, a label for the dimension, the marked increments of the scale, and the value increment. Both MarkedIncrement and

ValueIncrement are described by the attributes increment, startValue, and endValue. The DimensionIntersect is used when the scale contains more than one dimension. In its simplest form the attribute forAllDimensions is left at its default setting of “true” and the intersect point is defined by the attribute intersectValue. DisplayLayout is of CodeValueType and contains a definition of the type of scale such as scale line, value list, outline, etc.

EXAMPLES:

Sample question using the phrase completion method

I am aware of the presence of God or the Divine

Never **Continually**
0 1 2 3 4 5 6 7 8 9 10

This display has labels at the terminal anchors only and each increment is a Value Increment

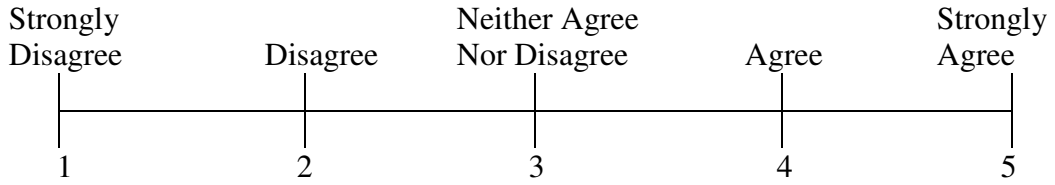
See ManagedRepresentationScheme / ScaleRepresentation (1) in example file (link at end of section)

Likert Item

The format of a typical five-level Likert item is:

1. Strongly disagree
2. Disagree
3. Neither agree nor disagree
4. Agree
5. Strongly agree

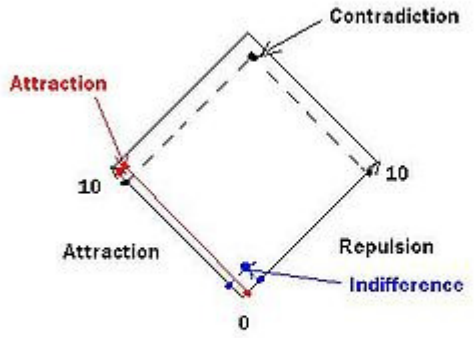
The above Likert Item is described and displayed as a CodeScheme. However if the display is scalar it could be described by a Scale.



This display has labels at all anchors and each increment is both a Marked Increment AND a Value Increment

See ManagedRepresentationScheme / ScaleRepresentation (2) in example file (link at end of section)

Diamond of Opposites



This particular display is an outline where the ends of the two intersecting scales form the corner points of the outline.

See ManagedRepresentationScheme / ScaleRepresentation (3) in example file (link at end of section)

MissingValuesRepresentationBase

Defines missing values as a numeric or code RepresentationBase which can be used with any other response domain or value representation. When combining a MissingValueRepresentationBase with valid responses or representations the user must take care not to replicate any valid response or representation. This structure allows for specifying missing values, specific definition of missing values through the use of a CodeRepresentationBase, and the definition of a blank (null) as a missing value.

```
MissingValuesRepresentationBase
   Extension base: VersionableType
   MissingValueRepresentationBaseName  (0..n)
   Label                               (0..n)
   Description                         (0..1)
   CHOICE                              (0..n)
      CodeRepresentationBase
      NumericRepresentationBase
      TextRepresentationBase
```

```
ENDCHOICE
GenerationInstructionReference      (0..1)
@isBlankMissingValue                default="true"
```

MissingValueRepresentationBase provides multiple means of describing missing values. They may contain any combination of a CodeRepresentationBase, NumericRepresentationBase, or TextRepresentationBase. In addition, the process of determining the how the missing value is assigned (generation instruction) may be referenced.

See ManagedRepresentationScheme / MissingValuesRepresentation (1) in example file (link at end of section)

A Missing Value RepresentationBase containing a set of coded missing value types, a numeric unlabeled value and information on how to treat a blank (white space).

Representation Examples (RepresentationExamples.xml)

Appendix: Change List (DDI 3.1 to 3.2)

See CHANGELIST_3_2.xlsx