Global Justice XML Data Model Naming and Design Rules

Draft Version 0.4, 23 August 2005

Editor:

Webb Roberts, Georgia Institute of Technology

Contributors:

Abstract:

This document specifies the data model, XML artifacts, and XML data for use with the Global Justice XML Data Model.

Status:

This document is an early draft of a specification for GJXDM-conformant XML components. This document is incomplete, and will undergo considerable revision before being approved for use. This document will be updated by GTRI in concert with the Global XML Structure Task Force (XSTF).

Please send comments on this specification to member of the XSTF.

Table of Contents

1. Introduction

This document specifies an information sharing framework based on the Global Justice XML Data Model (GJXDM) and World Wide Web Consortium (W3C) eXtensible Markup Language (XML) Schema. Sponsored by the Office of Justice Programs (OJP), U.S. Department of Justice (DoJ), the GJXDM is the result of a combined government and industry effort under the Global Information Sharing Initiative (Global). The mission of Global is to improve the administration of justice and protect the nation's public by promoting practices and technologies for the secure sharing of justice information.

In 2002, Global commissioned the Global XML Structure Task Force (GXSTF) to collect justice and public safety information requirements from various state and local government sources. Over 16,000 data requirements were identified and analyzed. These were reduced to approximately 2,200 unique data properties that were incorporated into about 500 data objects. These reusable GJXDM data components are then rendered in W3C XML Schema. The schemas are available to justice practitioners and developers on the OJP Information Technology Website at http://www.it.ojp.gov/jxdm.

W3C XML Schema was designed to enable information interoperability and sharing by providing a common language for describing data precisely. As its name implies, W3C XML Schema was also designed to be extensible. The constructs it defines are basic building blocks – baseline data types and structural components. Users apply these building blocks to describe more complex data types and structures – in effect, a new application with a domain-oriented syntax and vocabulary. This is what the GJXDM renders in XML Schema for the justice community. A reasonable set of rules and constraints on the employment of the building blocks and the GJXDM domain components will enhance information sharing by facilitating better interoperability. Therefore, this document specifies enforceable constraints for GJXDM schemas. It is a product of the GXSTF, the steering committee that continues to guide, manage, and approve changes to the GJXDM.

1.1. Audience

The primary audience for this document is the justice practitioners and developers who employ XML for information exchange and interoperability. The XML schemas rendered from the GJXDM still offer schema designers much flexibility and freedom to extend types and create new properties to satisfy requirements at the local level. However, these rules are intended to establish and, more importantly, enforce a degree of standardization at the national level.

1.2. The GJXDM XML Reference Architecture (In Brief)

The GJXDM is a reference model of unconstrained components rendered in XML Schema. Associated with the GJXDM schemas is an XML reference architecture that organizes and guides the employment of the various kinds of schemas that compose a GJXDM information exchange. The XML reference architecture is a visual representation of the relationships between XML schemas for GJXDM Information Exchange Package Documentation (IEPD) depicted in Figure 1. A GJXDM IEPD is a set of artifacts that describe an Information Exchange Package, a standard message structure as defined by the Federal Enterprise Architecture Consolidated Reference Model Document. Refer to the *Global JXDM Information Exchange Package Documentation Guidelines, Version 1.1* for a more detailed explanation of IEPDs and their contents.

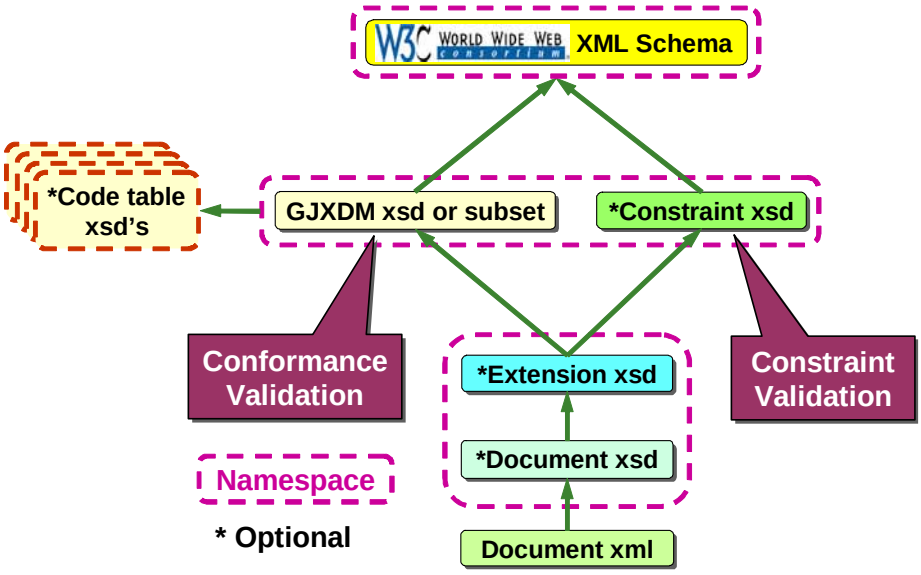

Figure 1. - GJXDM XML Reference Architecture

There are generally four categories of XML schemas used to specify the instances of a particular GJXDM information exchange:

- the GJXDM schemas (or a subset thereof),
- a constraint schema,
- an extension schema, and
- a document schema.

The latter three schemas are optional. The only mandatory schema is the GJXDM base schema or a correct subset of it (Subset schema derivation is defined in Section 7: Subset Schemas).The GJXDM schemas may import code table schemas (or subsets) as needed. An optional *document* schema imports, re-uses, and organizes the components from the GJXDM for the particular exchange. An optional *extension* schema may be used to add extended types and properties for components not contained in the GJXDM,.

The document and extension schemas can be combined into a single schema and namespace, or can be broken out into separate schemas and corresponding namespaces. The user may decide the best way to organize components. If the extension components will be reused elsewhere, it may be more efficient to maintain them in a separate namespace, rather than including them in a document namespace.

The GJXDM schemas are all inclusive and unconstrained. By creating a subset, the user can limit the components to only those he needs. Subsets can be created from the GJXDM base schema and code table schemas as well. The basic principle for a subset is that an instance that validates against a correct subset schema will always validate against the full GJXDM schema set. The user may also adjust cardinality constraints as desired within the subset schemas. Additional constraints can be handled in a *constraint* schema. A constraint schema may be derived from the subset schema, however, it can contain other constraints (for example, xsd:choice). The constraint schema provides a second *constraint validation* path that allows the user to reduce the possible set of correct XML instances independently from the GJXDM schema

or subset *conformance validation* path. This is done through multi-pass validation. A correctly constructed XML instance will validate through both the conformance and the constraint path.

1.3. Scope

This document is a specification for the GJXDM 3.1. It is not a specification for earlier versions of GJXDM, as earlier versions do not conform to 3.1 rules. It is not intended to specify beyond the GJXDM 3.1 release. The document addresses several issues:

- Definition of GJXDM-conformant schemas
- Definition of GJXDM-conformant reference schemas, on which schemas that are simply conformant are based
- Definition of subsetting methodology, through which conformant schemas are built from conformant reference schemas
- Naming of content to ensure understandability and reuse
- Documentation of content to ensure comprehension
- Definition of GJXDM-conformant instances, which contain additional validation requirements, such as types associated with references and relationships.

This document does not address the following:

- A formal definition of the data model. Such a definition would focus on RDF and concepts not strictly required for interoperability. The document instead focuses on definition of schemas that work with the data model, to ensure translatability and interoperability.
- Definition of versioning. The GJXDM distribution has a versioning mechanism in place, consisting of version numbers, with rules for what constitutes a "minor" or "major" change, and rules for inter-version compatibility. Such rules are not strictly required for peer-level interoperability, and will be added at a later stage.
- Definition of envelopes. This document does not define mechanisms related to the transport of GJXDM-conformant data between two points.

This document is intended as a technical specification. It is not intended to be a tutorial or instructional document.

1.4. Document Conventions

1.4.1. Logical Quoting

This document uses "logical quoting", in which, when required, exact terms are placed within quotes, with supporting punctuation placed outside the quotes. For example, when discussing a string with the value of "an exact value", we do not quote it as "an exact value," or as "an exact value." For such cases, we would use "an exact value", or "an exact value". In these cases, punctuation is placed outside the quotes, instead of within the quotes, as it would be with traditional quoting.

1.4.2. RFC 2119 Terminology

Within normative content (rules and definitions), the key words MUST, MUST NOT, REQUIRED, SHALL, SHALL NOT, SHOULD, SHOULD NOT, RECOMMENDED, MAY, and OPTIONAL in this document are to be interpreted as described in **[RFC2119]**.

1.4.3. XML Information Set Terminology

The following terms are used as defined by **[XMLInfoSet]**:

- Element parent
- Element child

  Note that the "child" of an element is a direct, immediate child. Children of an element, and their children, etc, will be referred to as "descendants" of that element.
- Document element

  The term "document element" is preferred over "root element".
- Attribute owner element

  An "owner element" is the element that possesses or contains the attribute.
- Attribute references

  The "references" value of an attribute is the list of elements referred to by the IDREFS or IDREF value of an attribute.

1.4.4. XML Schema Terminology

The terms “W3C XML Schema” and “XSD” are used throughout this document. They are considered synonymous; both refer to XML Schemas that conform to Parts 1 and 2 of the W3C *XML Schema Definition Language* (XSD) Recommendations (**[XMLSchemaStructures]** and **[XMLSchemaDatatypes]**).

The term "schema component" is defined by XSD. XML Schema contains specific definitions for various elements acting as particular types of schema components, including "model group definition schema component" and "Element declaration schema component". Such definitions are referred to, rather than restated.

1.4.5. Normative and Informative Content

The GJXDM NDR includes a variety of content. Some text is normative (binding in implementations), while other content is informative, including supporting text and specific rationale for rules. Some conventions used within the document include:

- [GJXDM 3.1 CHANGE]: The rule or principle in which this appears is considered a change from GJXDM 3.0.
- [Definition <*term*>]
  A formal definition of a term. Definitions are normative.
- [Principle <*number*>]
  A guiding principle for the GJXDM. The principles represent the requirements, concepts, and goals that have helped shape the GJXDM. Principles are informative, but act as the basis on which the rules are defined.
- [Rule <category><number>]
  A binding rule. The rules are normative. They should state how they bind the users. Most rules apply to conformant schemas (q.v.), while others apply to instances or reference schemas (q.v.).

  The rules are categorized, to make indexing simpler. Categories for rules are as specified in Table 1: Rule Categories.

Table 1: Rule Categories

Rule short name	Meaning
ATD	Attribute Definition Rules
ATN	Attribute Naming Rules
CSR	Constraint Schema Rules
CTD	Complex Type Definition Rules
DOC	Documentation Rules
GNR	General Naming Rules: Broadly-applicable rules for naming entities.
GXS	General XML Schema Rules
IND	Instance Document Rules
SSR	Subset Schema Rules
STA	Standards: The GJXDM's relation to standards, standards compliance, and interpretation of standards
STD	Simple Type Definition Rules
STR	Structures: The GJXDM's use of specific structural conventions to represent non-hierarchical data and object order.

Rule identifiers that are deleted or recategorized will not be reused until a major release milestone is reached, at which point all identifiers may be reset.

1.5. Syntax and Formatting

Courier: All words appearing in courier font are values, objects, and keywords.

Italics: All words appearing in italics, when not titles or used for emphasis, are special terms with definitions appearing in this document.

Keywords: keywords reflect concepts or constructs expressed in the language of their source standard. Keywords have been given an identifying prefix to reflect their source. The following prefixes are used:

- `xsd:` represents W3C XML Schema Definition Language. Note that use of the prefix "xsd" in schemas and instances is not required.
- `xsi:` represents W3C XML Schema's XML Schema Instance namespace. Note that use of the prefix "xsi" in schemas and instances is not required.
- `structures:` represents the GJXDM structures namespace. Note that use of the prefix "structures" in schemas and instances is not required.

Rules and supporting text may use Extended Backus-Naur Form (EBNF) notation as defined by **[XML]**.

See Appendix E: Glossary for additional term definitions.

2. Guiding Principles

Principles in this specification provide a foundation and explanations for the rules. The principles are not operationally enforceable. The rules are the normative and enforceable manifestation of the principles.

2.1. Specification Principles

Principles regarding what to specify, and what this document covers.

2.1.1. Minimal Specification

This specification should state what is required for interoperability, not all that could be specified. Certain decisions (such as normative XML comments) could create roadblocks for interoperability, making heavy demands on systems for very little gain. The goal is not standardization for standardization's sake. The goal is to maximize interoperability and reuse.

[Principle 1]	This specification should specify what is necessary for interoperability, and no more.

2.1.2. Schema-Level Specification

This specification should try, as much as is possible, to specify schema-level content. This is a specification for schemas, and so should specify schemas. It should avoid specifying complex data models, or data dictionaries.

[Principle 2]	This specification should focus on providing rules for specifying schemas.

2.1.3. Specificity and Conciseness

A rule should be as precise and specific as possible, to avoid broad, hard-to-modify rules. Putting multiple clauses in a rule makes it harder to modify. Using separate rules allows specifics of the spec to be clearly stated.

[Principle 3]	This specification should feature rules which are as specific, precise, and concise as possible.

2.2. Data Model Principles

The definition of the data model follows numerous guidelines. It is based upon actual data requirements gathered from a large number of exchanges in the justice domain, as well as a need to regularize data definitions to make them understandable and implementable.

[Principle 4]	GJXDM schemas and data instances are constructed in such a way as to maintain consistency of its fundamental data model.

2.2.1. RDF Data Model

The GJXDM data model is defined with the RDF data model at its core. The rules specified in this document ensure that GJXDM-conformant XML instances preserve the Subject-Property-

Object triplets defined by RDF. This support will allow for leveraging of Semantic Web and other higher-level understanding of data.

[Principle 5]	The GJXDM data model follows the Subject-Property-Object data model defined by RDF.

The RDF data model is defined by **[RDFConcepts]**.

2.2.2. Specialization of Types

The GJXDM embraces the fundamental concept of specialization of types. Through specialization, general concepts are made more precise for specific cases. Specialization of types involves the creation of new types by extending or restricting existing types.

[Principle 6]	Types are specialized through the use of derived types

2.2.3. Specialization of Properties

The specialization of properties involves basing narrow concepts on general concepts. Properties are described by **[RDFConcepts]** as characteristics or relationships. We represent them in XML as elements and attributes.

[Principle 7]	Properties are specialized through the use of derived properties

2.3. Principles in the use of XML

There are numerous methods and best practices for the use of XML.

2.3.1. Invariant Content

XML Schema has constructs that can make the data provided by XML processors different before and after schema processing. A sample of this is the use of attributes with default values. Before processing, there may be no attribute value, but after processing, the attribute value exists.

Within the GJXDM, the process of validation of instances against schemas is solely validation: testing that data instances match desired constraints and guidelines. It should not be used to change the content of data instances.

[Principle 8]	The content of a data instance must not be modified by processing against schemas.

2.3.2. Order of Data

[Principle 9]	The order of data in an XML instance should not be assumed to be the natural order of the information.

XML data, when not used for "markup" should be interpreted in a fashion much like a structured type in a programming language. The fields in a structured type have an order, but that order is not assumed to be the order of the content. A type may contain a field "Foo", "Bar", and "Baz", but that does not mean that the value contained by "Bar" comes after the value in "Foo" and before the value in "Baz". For such requirements, sequence should be defined explicitly.

2.3.3. XML Schema for Validation

The GJXDM is designed for W3C XML Schema validation. A primary goal is to maximize the amount of validation that may be performed by XML Schema validating parsers.

[Principle 10]	The GJXDM should depend on W3C XML Schema validating parsers for validation of XML content.

Keep in mind that XSD validates content using content models: descriptions of what elements and attributes may be contained within an element and what values are allowable. Mechanisms involving linking using attribute and element values are useful, but should only be relied upon when absolutely necessary.

2.3.4. Minimal implementation requirements

The GJXDM is intended to be an open specification, supported by many diverse implementations. It was designed from data requirements and not from or for any particular system or implementation. Use of the GJXDM should not depend on specific software, other than XML Schema validating parsers.

[Principle 11]	The GJXDM should not depend on specific software packages, frameworks, or systems for interpretation of XML instances.

Similarly, the GJXDM should be implementable with commercial off-the-shelf and free software products.

[Principle 12]	The GJXDM should be implementable with a variety of commercial off-the-shelf and free software products.

2.3.5. Reference Schema Defines Namespace Contents

The GJXDM uses the concept of a *reference schema*, which defines the structure and content of a namespace. For each namespace, there is exactly one reference schema. A user may use a subset schema (q.v.) for a reference schema, but all instances must validate against a single reference schema for each namespace.

[Principle 13]	Each namespace used with the GJXDM will be defined by exactly one reference schema

2.3.6. Reuse of Namespaces

The GJXDM is designed to maximize reuse of namespaces and the schemas that define them. When referring to a concept defined by the GJXDM, users should ensure that instances and schemas refer to the namespace defined by the GJXDM. User-defined namespaces should be used for specializations and extension of GJXDM constructs, but should not be used when the GJXDM structures are sufficient.

[Principle 14]	GJXDM-conformant instances and schemas should reuse the GJXDM namespaces when possible.

Reuse is by reference to the namespace, with validation against reference schemas or reference subset schemas.

2.3.7. Specific Typing

As soon as an application has determined the name and namespace of an attribute or element used in GJXDM-conformant instances, it will also know the type of that attribute or element. GJXDM does not employ anonymous typing.

[Principle 15]	Each attribute and element within the GJXDM has a defined type.

2.3.8. Avoidance of Wildcards

Wildcards in schemas work in opposition to standardization. The effort of creating harmonized, standard schemas is to standardize a set of data. Wildcards allow non-standard data to be passed in otherwise standard messages. As such, users may receive non-standard data, and users may not be encouraged to extend in such a way that extensions may be distinguished from standardized content.

[Principle 16]	Wildcards in standard schemas should be avoided

2.3.9. Schema Location as a Hint

[XMLSchemaStructures] specifies schemaLocation, an attribute of the xsd:import element of a schema, the xsi:schemaLocation, and xsi:noNamespaceSchemaLocation attributes of XML instances. In both of these uses, the specification explicitly maintains that the schema location specified is a hint, which may be overridden by applications. For example, from **[XMLSchemaStructures]**:

> The actual value of the schemaLocation, if present, gives a hint as to where a serialization of a schema document with declarations and definitions for that namespace (or none) may be found.

[Principle 17]	Schema locations specified within schemas and XML instances are hints, to provide default values to processing applications.

2.3.10. Multi-pass Validation

Systems that operate on XML data have the opportunity to perform multiple layers of processing. Data may be processed by middleware, XML libraries, XML Schemas, and application software.

[Principle 18]	The primary purpose of XML Schema validation is to restrict processed data to that data that conforms to agreed-upon rules. This restriction is achieved by marking as invalid that data that does not conform to the rules defined by the schema.

The GJXDM does not attempt to create a one-size-fits-all schema, to perform all validation. Instead, it creates a set of reference schemas, on which additional constraints may be placed. It also does not focus on language-binding XML Schema implementations, which convert XSD definitions into working executables. It is, instead, focused on normalizing language and preserving the meaning of data.

[Principle 19]	Constraints on XML instances MAY be validated by multiple schema validation passes, using multiple schemas for a single namespace.

2.3.11. No Mixed Content

When validating XML instance data against W3C XML Schemas, mixed content is very difficult to constrain. Instances that use mixed content are difficult to specify, and complicate the task of data processing. Much of the payload carried by mixed content is unchecked, and does not facilitate data standardization or validation.

[Principle 20]	The GJXDM does not specify data that uses mixed content.

2.3.12. Application versus User Information

[Principle 21]	XML data is primarily intended for automatic processing, not for human consumption.

XML should be made human-understandable whenever possible, but it is not targeted at human consumers. XML Schema is intended for validators and automatic processing. HTML is intended for browsers. Browsers and similar technology provide human interfaces to XML and other structured content. As such, structured XML content does not belong in places targeted towards human consumption. Human-targeted information should be of a form suitable for presentation.

2.4. Design for Extensibility

The GJXDM is designed to be extended. Numerous methods are considered acceptable in creating extended and specialized components.

[Principle 22]	The GJXDM is intended for extension and augmentation by users and developers outside the standardization process.

3. Relation to Standards

The GJXDM uses many public standards, and is influenced by many others. This section specifies to what specifications the GJXDM conforms, and the specific rationale for differences from public standards.

3.1. XML 1.0

[Rule STA1]	GJXDM-conformant schemas and instances MUST conform to XML as specified by **[XML]**.

3.2. XML Namespaces

[Rule STA2]	GJXDM-conformant schemas and instances MUST conform to the specification for namespaces in XML, as defined by **[XMLNamespaces]** and **[XMLNamespacesErrata]**.

3.3. XML Schema

The W3C XML Schema definition language has become the generally accepted schema language that is experiencing the most widespread adoption. Although other schema languages exist that offer their own advantages and disadvantages, the best approach is to base GJXDM on W3C XML Schema.

[Rule STA3]	All GJXDM-conformant schemas MUST be based on the W3C XML Schema Recommendations: XML Schema Part 1: Structures and XML Schema Part 2: Datatypes, as specified by **[XMLSchemaStructures]** and **[XMLSchemaDatatypes]**.

Rationale	This document is to be the specification for schemas and instances, not a specification for the specification itself. Those go in principles.

3.4. `xml:id`

The xml:id specification provides a normalized identifier, for various XML processors to be able to recognize XML identifiers unambiguously. Other methods include the use of the XSD type "`ID`", which is only available to XML Schema processors.

[Rule STA4]	[GJXDM 3.1 CHANGE] GJXDM-conformant schemas and instances MUST conform to the specification for xml:id, as specified by **[XML-ID]**.

Rationale	The xml:id specification provides a normalized, recognizable, unambiguous attribute for an XML ID.

3.5. ISO 11179

3.5.1. ISO 11179, Part 4

3.5.1.1. Formulation of data definitions

The ISO 11179, Part 4, standard provides structure and rules for defining data definitions. The GJXDM uses this standard with respect to summary definitions.

[Rule STA5]	Within a GJXDM-conformant schema, each XML element, attribute, and type summary definition SHALL follow the rules and recommendations of formulating data definitions given by ISO 11179, Part 4.

Rationale	To advance the goal of creating semantically-rich GJXDM conformant schemas, it is necessary that data definitions be descriptive and well-formed.

Note that GJXDM full descriptions may contain extensive details about an XML element, attribute, or type, including such things as a rationale, examples, and domain-specific usages. As such, they are not bound to the rules and recommendations of ISO 11179, Part 4.

3.5.2. ISO 11179, Part 5

The ISO 11179, Part 5, standard provides a structure and rules for naming data elements. The GJXDM uses this standard, with some specific refinements.

3.5.2.1. Object Class

In the GJXDM, the *object class* that constitutes the first part of an entity name is interpreted as a real-world object class. That is, the object class term should reflect the real-world object classes and not specific data classes. It represents a real-world object rather than simply a collection of data.

3.5.2.2. Representation Terms

[Rule GNR1]	Each XML element, attribute, and type defined by GJXDM-conformant schemas SHALL use a representation term from Table 2: Representation Terms unless the XML elements are of types with complex content.

Rationale	A representation term defines the kind of value that is to be expected from the element. It is not needed for elements that are of types with complex content because they are comprised of other elements. There is no single kind of value to be expected within an element of complex content.

[Rule GNR2]	GJXDM-conformant schemas SHALL use the representation term "Type" in the name of each non-enumerated XML type.

Rationale	Using the representation term "Type" immediately identifies XML types in a GJXDM-conformant schema and prevents naming collisions with corresponding XML elements and attributes.

[Rule GNR3]	GJXDM-conformant schemas SHALL use the representation term “CodeType” in the name of each enumerated XML type.

Rationale	Using the representation term "CodeType" immediately identifies XML types in a GJXDM-conformant schema that define code sets and prevents naming collisions with corresponding XML elements and attributes.

Table 2: Representation Terms

Representation Term	Definition
Amount	A number of monetary units specified in a currency where the unit of currency is explicit or implied.
BinaryObject	A set of finite-length sequences of binary octets.
Graphic	A diagram, graph, mathematical curves, or similar representation
Picture	A visual representation of a person, object, or scene
Sound	A representation for audio
Video	A motion picture representation; may include audio encoded within
Code	A character string (letters, figures or symbols) that for brevity, language independence, or precision, represents a definitive value of an attribute.
CodeText	A character string for which data values are codes but are not validated by the schema because there is no corresponding enumerated type present.
DateTime[1]	A particular point in the progression of time together with relevant supplementary information.
Date	A particular day, month, and year in the Gregorian calendar.
Time	A particular point in the progression of time within an unspecified 24 hour day.

[1] DateTime is not actually used in the GJXDM reference distribution schema, but is available for use.

Representation Term	Definition
DescriptionText	A character string that is a description of a value, not the actual value itself.
Identifier[2]	A character string to identify and distinguish uniquely, one instance of an object in an identification scheme from all other objects in the same scheme together with relevant supplementary information.
Indicator	A list of two mutually exclusive Boolean values that express the only possible states of a Property.
Measure	A numeric value determined by measuring an object along with the specified unit of measure.
Numeric	Numeric information that is assigned or is determined by calculation, counting, or sequencing. It does not require a unit of quantity or unit of measure.
Value	A result of a calculation
Rate	A representation of a ratio where the two units are not included.
Percent	A representation of a ratio in which the two units are the same.
Quantity	A counted number of non-monetary units possibly including fractions.
Text	A character string (i.e. a finite sequence of characters) generally in the form of words of a language.
Name	A word or phrase that constitutes the distinctive designation of a person, place, thing or concept.
Type	The expression of the aggregation of properties to indicate the aggregation of lower leveled information entities. All Type names use this Representation Term

[2] "ID" (the abbreviation) is preferred over the full term "Identifier". It is indicated in the table of abbreviations.

4. Normalized Structure

4.1. `xml:id`

[Rule STR1]	[GJXDM 3.1 CHANGE] Within GJXDM-conformant schemas, a complex type SHALL NOT be defined with an attribute other than `xml:id` as an ID attribute as defined by **[XML]**.

Rationale	Schemas need not include an ID attribute, but if they do, it needs to be xml:id.

4.2. Structures Namespace

The GJXDM provides a namespace containing structures for organizing data. These structures should be used to augment XML data. The structures provided are not meant to replace fundamental XML organization methods; they are intended to assist them.

[Rule STR2]	[GJXDM 3.1 CHANGE] The GJXDM structures namespace shall be represented by the URI "`http://www.it.ojp.gov/jxdm/structures/1`".

Rationale	The structures namespace is a single namespace, separate from the GJXDM data content. This document refers to this content via the prefix "structures:".

[Rule STR3]	[GJXDM 3.1 CHANGE] GJXDM-conformant schemas and instances SHALL NOT use content within the GJXDM structures namespace except as specified by this document.

Rationale	It is an error to put into the GJXDM structures namespace types, elements, attributes, etc., that are not specified by this NDR.

4.2.1. Sequence ID

In keeping with the principle that order within an XML instance is not necessarily natural order, the attribute structures:sequenceID is provided to allow specification of natural order. An example would be for proper names, where the natural order of the names may not appear in the same order as the sequence defined by a complex type. The sequenceID attribute allows instances to express natural order of data relative to a parent. The order of data is as yielded by XSLT's sort element, with data-type of "number", and order of "ascending". Content with identical sequenceID values has undefined order.

[Rule STR4]	GJXDM-conformant schemas and instances SHALL NOT be interpreted such that the order of data within an XML instance is understood to be the order of the referents. Sequence of data within an XML instance is meaningless.

Rationale	Because of GJXDM's use of structured, defined types, and its use of sequence, as well as various representation mechanisms, the order of data within an XML instance has no meaning. True order of objects (such as parts of a name, or lines in an address, or parts of a phone number) must use an explicit method to define their order.

[Rule STR5]	[GJXDM 3.1 CHANGE] Within a GJXDM-conformant schema, sequential order of referents, when indication of such is necessary, MUST be indicated via the use of the attribute structures:sequenceID.

Rationale	When order of objects must be specified, the sequenceID attribute must be used. This does not apply to objects which are ordered according to some actual value, such as date, or size, or distance. This applies to simple sequential order, such as components of a name, or sequence of items in a list.

[Rule STR6]	Within GJXDM-conformant schemas and instances, the attribute structures:sequenceID SHALL NOT be interpreted as meaningful beyond an indicator of sequence of objects relative to its siblings.

Rationale	Siblings of a data item are items that have the same parent. Note that, using the reference and relationships mechanisms, data objects may have multiple parents. The sequenceID is truly metadata, helping to express the structure of the data, rather than its content.

[Rule STR7]	Within GJXDM-conformant schemas, the order of objects SHALL be given by sorting the objects by numerical value of their respective attribute `structures:sequenceID`, from smallest to highest. The relative order of objects with equal values for `structures:sequenceID` is undefined. The order of objects with no value for `structures:sequenceID` is undefined.

Rationale	Links between objects are ordered by the value of sequenceID. Objects with identical or no value for sequenceID are ambiguous.

4.2.2. References

[Rule STR8]	Within a GJXDM-conformant schema, a *reference element* is an element defined with a name of the form NCName "`Reference`" ("`.`" NCName)? Where NCName is as defined by **[XMLNamespaces]**.

Rationale	Reference elements allow XML data to break free of the hierarchical data model, allowing reuse of data objects.

[Rule STR9]	[GJXDM 3.1 CHANGE] Within a GJXDM-conformant schema, a reference element SHALL be defined to be of type `structures:ReferenceType`.

Rationale	Reference elements must be of the reference type.

[Rule STR10]	Within a GJXDM-conformant schema, the *fundamental element* of a reference element with a name of the form NCName1 "`Reference.`" NCName2 is an element with a name of NCName1 "." NCName2

	Where the values of NCName1 and NCName2 are consistent between the two element names. NCName1 and NCName2 match NCName as defined by **[XMLNamespaces]**.

Rationale	A fundamental element is the data-carrying element on which the reference element is based.

[Rule STR11]	Within a GJXDM-conformant schema, the *fundamental element* of a reference element with a name of the form NCName "`Reference`" is an element with a name of NCName Where the value of NCName is consistent between the two names. NCName is as defined by **[XMLNamespaces]**.

Rationale	Syntax for element names are simpler without the use of a representation qualifying suffix.

[Rule STR12]	A GJXDM-reference schema that contains a reference element definition MUST contain the corresponding fundamental element definition.

[Rule STR13]	Within GJXDM-conformant schemas, an element defined with a name not of the form defined in [Rule STR8] SHALL NOT be of type `structures:ReferenceType`.

Rationale	If an element is not named to be a reference element, then it may not be of reference type. Only reference elements may be of reference type.

[Rule STR14]	[GJXDM 3.1 CHANGE] Within a GJXDM-conformant instance, an element of type `structures:ReferenceType` MUST contain an occurrence of attribute `structures:reference` that satisfies the constraints on values of type IDREF as specified by **[XML]**.

Rationale	ReferenceType contains a reference, which must be a proper IDREF.

[Rule STR15]	Within GJXDM-conformant schemas, a defined type MUST NOT use the type `structures:ReferenceType` as a base for extension or restriction.

Rationale	ReferenceType stands alone. It is not intended for extension.

[Rule STR16]	Within a GJXDM-conformant instance, an element of type `structures:ReferenceType` MAY contain an occurrence of attribute `xml:id`.

Rationale	References may have IDs.

[Rule STR17]	Within a GJXDM-conformant instance, the element referred to by an attribute `structures:reference` MUST be of a type valid for the object of the fundamental element of the reference element.

5. Schema Naming and Design Rules

[Definition GJXDM-conformant schema]
The term *GJXDM-conformant schema* SHALL be defined as an XML Schema that complies with the rules for GJXDM-conformant schemas as defined by this specification.

Rationale: This specification is primarily concerned with defining a particular type of schema that is designed to match the numerous requirements and principles specified in Section 2: Guiding Principles.

[Definition GJXDM-conformant reference schema]
The term *GJXDM-conformant reference schema* SHALL be defined as an XML Schema that complies with the rules for GJXDM-conformant reference schemas as defined by this specification.

Rationale: This specification separates reference schemas from non-reference schemas. Reference schemas are the fully-documented forms of schemas that contain all available content, to the largest available cardinality.

These reference schemas may act as the basis for subset schemas, which are not reference schemas, and which may apply certain constraints, restrictions, and narrowing of scope to the reference schema.

Also included in this specification is the concept of constraint schemas. Constraint schemas act in tandem with the reference schemas, and act to restrict content specified by the reference schemas. Constraint schemas need not be GJXDM-conformant, as the content on which they act must also validate against conformant schemas. Such validation may be performed in stages, in agreement with the principle of multiple-pass validation.

5.1. General Schema Naming and Design Rules

The W3C XML Schema language provides many redundant features that allow a developer to represent a logical data model many different ways. Heterogeneous data models can become an interoperability problem in the absence of a comprehensive set of naming, definition, and declaration design rules.

This section establishes rules for XML schema elements, attributes, and type creation. Because the W3C XML specifications are flexible, comprehensive rules are needed to achieve a balance between establishing uniform schema design while still providing developers flexibility across the Justice and Public Safety domain.

5.1.1. Naming of Entities

5.1.1.1. Usage of English

The English language has many spelling variations for the same word. For example, American English “program” has a corresponding British spelling “programme.” This variation has the potential to cause interoperability problems when exchanging XML components because of the different names used by the same elements. Providing a dictionary standard for spelling will mitigate this potential interoperability issue.

[Rule GNR4]	GJXDM information exchange XML elements, attributes and type names MUST be composed of words from the English language, using the prevalent U.S. spelling, as provided by the Oxford English Dictionary, Second Edition, 1989.

5.1.1.2. Characters in Names

[Rule GNR5]	GJXDM information exchange XML element, attribute and type names SHALL use only the characters from Table 3: Characters Allowed in Names, in accordance with the use specified in that table.

Names of entities within the GJXDM follow the rules of W3C XML Schema, by rule [Rule STA3]. Entities also must follow the rules specified for each type of XML Schema entity.

Table 3: Characters Allowed in Names

Character Title	Character Literal	Use
Letters		The first character of a name SHALL be a letter.
Uppercase letters	'A'-'Z'	The first character of the name of a type or an element SHALL be an uppercase letter
Lowercase letters	'a'-'z'	The first character of the name of an attribute SHALL be a lowercase letter
Digits	'0'-'9'	Digits SHALL NOT be used to enumerate. They may be used to specify a specific concept or standard.
Underscore	'_'	Underscores SHALL NOT be used in GJXDM entity names
Hyphen	'-'	Hyphens may be used in the Representation Qualification Suffix portion of an element name
Period	'.'	Periods may be used to separate a property name from its Representation Qualification Suffix.

5.1.1.3. Use of Acronyms and Abbreviations

Acronyms and abbreviations can obscure meaning, and impair understanding and interoperability. They should be used with great care. Acronyms and abbreviations that are used must be documented, and used consistently.

[Rule GNR6]	A GJXDM-conformant schema MUST consistently use approved acronyms, abbreviations, and word truncations within defined names. The approved shortened forms are defined in Appendix B: Normative Abbreviations.

Other acronyms and abbreviations will be used on a per-schema basis. Such abbreviations must be properly documented within the schema documentation.

[Rule DOC1]	[GJXDM 3.1 CHANGE] A GJXDM-conformant schema MUST specify ALL acronyms, abbreviations, and other word truncations within GJXDM-conformant schema notation.

5.1.1.4. Singular and Plural Forms

[Rule GNR7]	Within GJXDM-conformant schemas, element, attribute and type names MUST be in singular form unless the concept itself is plural.

The following is an example of correct name use:

```
PersonPhysicalFeature, PhysicalFeatureType
PersonPhysicalDetails, PersonPhysicalDetailsType
personNameInitialIndicator
```

5.1.1.5. Character Case

[Rule GNR8]	The upper camel case convention SHALL be used for naming elements and types.

Rationale	The use of upper camel case for names of types has become a defacto standard, to which GJXDM conforms.

Examples of upper camel case names:

```
PersonName
JewelryStone
```

[Rule GNR9]	The names of attributes defined within GJXDM-conformant schemas SHALL be formatted in lower camel case.

Examples of lower camel case names:

```
amountCurrencyCodeListVersionID
characterSetCode
```

5.1.1.6. Mixed Content

[Rule CTD1]	GXDM-conformant schemas SHALL NOT define XML elements that contain mixed content.

Rationale	GJXDM does not support mixed content in XML elements. Exchange documents containing mixed content are difficult to process, define, and constrain.

5.1.2. Notations

Notations are not supported by the GJXDM. Notations allow the attachment of system and public identifiers on fields of data.

[Rule GXS3]	GJXDM-conformant schemas SHALL NOT contain an occurrence of the element `xsd:notation`.

Rationale	The notation mechanism is not supported by GJXDM. The xsd:notation element defines a notation on a field of data.

[Rule GXS4]	GJXDM-conformant schemas SHALL NOT contain a reference to the type `xsd:notation`, or to a type derived from that type.

Rationale	The notation mechanism is not supported by GJXDM. The xsd:notation type defines a field to which system and public identifiers may be applied.

5.2. Schema Document Element

The features of W3C XML Schema allow for flexibility of use for many different and varied types of implementation. The GJXDM NDR requires consistent use of these features. The document element of a schema is named schema, from the namespace `http://www.w3.org/2001/XMLSchema`, as specified by **[XMLSchemaStructures]**.

[Rule GXS5]	In a GJXDM-conformant schema, any occurrence of the element `xsd:schema` MUST own an attribute `targetNamespace`.

Rationale	Schemas without defined namespaces provide definitions that are ambiguous, in that they are not universally identifiable.

[Rule GXS6]	In a GJXDM-conformant schema, the value of any occurrence of the attribute `targetNamespace` owned by an element `xsd:schema` MUST match the production `<absolute-URI>` as defined by **[RFC3986]**.

Rationale	Absolute URIs are the only universally meaningful URIs. Finding the targetNamespace using XML Base is overly complicated, and not specified by XSD. Relative URIs aren't universally identifiable

The xsd:schema element contains an optional attribute attributeFormDefault. The value of this attribute is immaterial to a GJXDM-conformant schema, as each attribute defined by a GJXDM-conformant schema must be defined at the top-level, and so must be qualified with the target namespace of its declaration.

The xsd:schema element contains an optional attribute elementFormDefault. The value of this attribute is immaterial to a GJXDM-conformant schema, as each element defined by a GJXDM-conformant schema must be defined at the top-level, and so must be qualified with the target namespace of its declaration.

[Rule GXS7]	In a GJXDM-conformant schema, there MUST NOT be an occurrence of the attribute `blockDefault` within an element `xsd:schema`.

Rationale	To make processing more uniform, the default value for blocking of substitutions should be left alone, and blocked entities should be specified explicitly, on an individual basis.

[Rule GXS8]	In a GJXDM-conformant schema, there MUST NOT be an occurrence of the attribute `finalDefault` within an element `xsd:schema`.

Rationale	To make processing more uniform, the default value for preventing derived types should be left alone, and final types should be specified explicitly, on an individual basis.

[Rule GXS9]	A GJXDM-conformant schema MUST NOT use the element `xsd:include`.

Rationale	Inclusion of namespaced schemas violates the principle that a single reference schema defines a namespace. It breaks a namespace up into arbitrary partial schemas, which needlessly complicates the schema structure. Inclusion of unnamespaced schemas complicates schema understanding as well, making it difficult to find the realization of a specific schema artifact.

[Rule GXS10]	A GJXDM-conformant schema MUST NOT use the element xsd:redefine.

Rationale	Use of redefine provides an alternative definition for the contents of a namespace, in violation of the principle that a single reference schema defines a namespace.

5.3. Import of Namespaces

Namespaces used by a GJXDM-conformant schema must be imported using the xsd:import element, in compliance with the XML Schema specification. Importing of namespaces is performed via the `xsd:import` element, which appears as an immediate child of the `xsd:schema` element.

[Rule GXS11]	Within a GJXDM-conformant schema, any occurrence of the element `xsd:import` MUST own the attribute `namespace`.

Rationale	An import that does not specify a namespace is de facto enabling reference to foreign components without a namespace, which impairs interoperability.

[Rule GXS12]	In a GJXDM-conformant schema, the value of any occurrence of the attribute `namespace` owned by an element `xsd:import` MUST match the production `<absolute-URI>` as defined by **[RFC3986]**.

Rationale	It is important that the namespace declared by a schema be universally defined, and unambiguous. XML Base processing is not specified by XML Schema, and so is not supported here.

[Rule GXS13]	Within a GJXDM-conformant schema, any occurrence of the element `xsd:import` MUST own the attribute `schemaLocation`.

Rationale	An import that does not specify a schema location gives no clue to processing applications as to where to find an implementation of the namespace. Even though such a provided schema location may be overridden, it is important that an initial default be provided for processing.

[Rule GXS14]	In a GJXDM-conformant schema, the value of any occurrence of the attribute `schemaLocation` owned by an element `xsd:import` MUST match either the production `<absolute-URI>`, or the definition of "*relative-path reference*", as defined by **[RFC3986]**.

Rationale	Default schemas must be provided for processing. These may specified either as absolute or relative URIs. Since URNs are not resolvable, they are inappropriate for use in schemaLocation. The requirement for conformance to "*relative-path reference*" is required to avoid the more obscure syntax of "*network-path reference*" and the system-specific "*absolute-path reference*".

[Rule GXS15]	In a GJXDM-conformant schema, the value of any occurrence of the attribute `schemaLocation` owned by an element `xsd:import` MUST be resolvable to a XML schema document file that is valid according to **[XMLSchemaStructures]** and **[XMLSchemaDatatypes]**

Rationale	The object imported via xsd:import must be a schema document. The XSD spec requires that the "author warrants" that this is the case. This rule ensures that this is actually the case.

[Rule GXS16]	Within a GJXDM-conformant reference schema, any occurrence of the element `xsd:import` MUST have an occurrence of the element `xsd:annotation` as an immediate child.

Rationale	Reference schemas must be properly documented.

5.4. General Type Definitions

Since GJXDM document and extension schema elements and types are intended to be reusable, all types must be named. This permits other types to establish elements that reference these types, and also supports the use of extensions for the purposes of versioning and customization.

The requirement that types be named is established by [Rule STD1] and [Rule CTD2].

GJXDM-conformant schemas may not use xsd:anyType, because this feature permits the introduction of potentially unknown types into an XML instance. GJXDM intends that all constructs within the instance be described by the schemas describing that instance – xsd:anyType tends to work counter to the requirements of interoperability. In consequence, particular attention is given to the need to enable meaningful validation of the GJXDM document instances.

[Rule GXS17]	GJXDM-conformant schemas SHALL NOT reference the type `xsd:anyType`.

Rationale	The type xsd:anyType provides a substantial wildcard by which untyped and unconstrained data may be carried. This violates several GJXDM principles.

5.5. Simple Type Definitions

[Rule STD1]	Within a GJXDM-conformant schema, any occurrence of the element `xsd:simpleType` MUST appear as an immediate child of the element `xsd:schema`.

Rationale	GJXDM does not support anonymous / unnamed types in conformant schemas. All "top-level" types are required by XSD to be named, and are therefore globally reusable.

[Rule STD2]	Within GJXDM-conformant schemas, any occurrence of the element `xsd:simpleType` MUST have an occurrence of the element `xsd:restriction` as an immediate child.

Rationale	Any simple type must be a restriction of another type. One alternative is "list", in which the resulting type is a list of entries, which should be structured via explicit XML, and not composed fields. The other alternative is "union", which combines different-typed entries into a single type, obscuring meaning of instance values.

[Rule STD3]	Within a GJXDM-conformant reference schema, any occurrence of the element `xsd:simpleType` MUST have an occurrence of the element `xsd:annotation` as an immediate child.

Rationale	Reference schemas must be properly documented.

[Rule STD4]	Within a GJXDM-conformant schema, any occurrence of the element `xsd:restriction` that is an immediate child of an element `xsd:simpleType` MUST contain an attribute `base`.

Rationale	All restrictions must restrict named types. GJXDM does not support anonymous types.

Taking into account the other rules, this rule may be derivable, however, it is useful to have the point stand on its own.

[Rule STD5]	Within a GJXDM-conformant schema, the value of the attribute `base` owned by an element `xsd:restriction` acting as part of a simple type declaration schema component MUST have a value that refers to a simple type defined by the XML Schema specification, or a simple type defined by a GJXDM-conformant schema.

The content of the simple type definition then may add facets to the base simple type, in line with XSD specifications.

[Rule STD6]	Within a GJXDM-conformant schema, the value of the attribute `base` owned by an element `xsd:restriction` acting as part of a simple type declaration schema component MUST NOT have a value that refers to `xsd:anySimpleType`

Rationale	xsd:anySimpleType is insufficiently constrained to provide a meaningful starting point for content definitions.

5.6. Complex Type Definitions

[Rule CTD2]	Within GJXDM-conformant schemas, any occurrence of the element `xsd:complexType` MUST appear as a child of the element `xsd:schema`.

Rationale	GJXDM does not support anonymous / unnamed types in conformant schemas. All "top-level" types are required by XSD to be named, and are therefore globally reusable.

[Rule CTD3]	Within GJXDM-conformant schemas, an occurrence of the element `xsd:complexType` MUST NOT include the attribute `mixed` with a value of "`true`" or "`1`".

Rationale	GJXDM does not support mixed content.

GJXDM supports use of attributes and attribute groups.

5.6.1. Complex Content

Within `xsd:complexType`, GJXDM supports use of `abstract`, `block`, and `final`. GJXDM supports use of simple content, complex content, `xsd:choice`, groups, sequences, attributes, and attribute groups.

[Rule CTD4]	Within a GJXDM-conformant schema, the element `xsd:all` SHALL NOT occur.

Rationale	GJXDM does not support use of `xsd:all`. Use of concretely-sequenced elements within complex types simplifies many types of processing, and allows reference schemas to act as a base for highly constrained, yet interoperable, subset schemas.

[Rule CTD5]	Within a GJXDM-conformant schema, an occurrence of the element `xsd:group` acting as a particle schema component according to **[XMLSchemaStructures]** MUST have values of "`1`" for the attributes `minOccurs` and `maxOccurs`.

Rationale	Cardinality is restricted to maintain the simple-sequence compatibility of complex content. GJXDM does not permit complicated patterns of interlacing of elements. Elements have a strict sequential occurrence.

The value of "1" for minOccurs and maxOccurs is provided as default by XSD, and so need not be explicitly expressed.

[Rule CTD6]	Within a GJXDM-conformant schema, an occurrence of the element `xsd:choice` MUST have values of "`1`" for the attribute `minOccurs` and `maxOccurs`.

The value of "1" for minOccurs and maxOccurs is provided as default by XSD, and so need not be explicitly expressed.

[Rule CTD7]	Within a GJXDM-conformant schema, an occurrence of the element `xsd:sequence` MUST have values of "`1`" for the attributes `minOccurs` and `maxOccurs`.

The value of "1" for minOccurs and maxOccurs is provided as default by XSD, and so need not be explicitly expressed.

[Rule CTD8]	Within a GJXDM-conformant schema, complex content SHALL NOT declare occurrences of a single element using more than one element statement.

Rationale	The goal here is simple sequences of elements. Allowing multiple element statements for a single element creates situations where "Foo" is followed by "Bar" and again by "Foo", which puts structural and organizational constraints within the XML data file.

[Rule CTD9]	Within a GJXDM-conformant schema, an element or attribute that is eliminated through restriction and reinserted by extension MUST conform to the original definition.

Rationale	The derived and extended content must maintain the "is-a" nature of derivation: Derived type "Foo" is-a base type "Bar". Any constraints on "Bar" must be maintained in the derived type "Foo".

5.6.2. Exclusion of Wildcards

[Rule CTD10]	GJXDM-conformant schemas SHALL NOT contain an occurrence of the element `xsd:anyAttribute`.

The element `xsd:anyAttribute` may appear within constraint schemas.

[Rule CTD11]	GJXDM-conformant schemas SHALL NOT contain an occurrence of the element `xsd:any`.

Rationale	The elements xsd:anyAttribute and xsd:any provide wildcards, which may carry undefined content, in violation of the principle of avoidance of wildcards.

The element `xsd:any` may appear within constraint schemas.

5.7. Element and Attribute Definitions

[Rule ATN1]	Each XML element and attribute name defined by the GJXDM MUST correspond to a single representation type.

Rationale	The name of a XML element or attribute from a GJXDM-conformant schema should be concrete. The element or attribute name alone should be sufficient in determining not only the semantic meaning, but also the type structure of that element or attribute.

[Rule ATN2]	XML element and attribute names MUST contain a representation qualifying suffix when a single semantic meaning has multiple representation types.

[Rule ATN3]	A representation qualifying suffix is a term that identifies a representation type.

Rationale	Rationale A single semantic concept will result in a single element or attribute name that may contain an object class term, a qualifier term, a property term, and a representation term. When a single semantic concept has multiple representation types available (each corresponding to the same representation

	term), an additional qualifier is necessary to make the element or attribute names unique. This additional qualifier is the representation qualifying suffix.

Example:

```
LocationCountryCode.iso3166Alpha2
LocationCountryCode.iso3166Alpha3
LocationCountryCode.iso3166Numeric
LocationCountryCode.fips10-4
```

These four elements represent the same semantic concept - a code identifying a location's country. "Location" is the object class term, "Country" is the property term, "Code" is the representation term. An additional term is necessary to identify the code set being used.

"`iso3166Alpha2`", "`iso3166Alph3`", "`iso3166Numeric`", and "`fips10-4`" are each representation qualifiers. The first three distinguish different code sets from the same source, ISO 3166. The fourth identifies a code set from the FIPS 10-4 source.

Note that in cases where there is one representation type that may be considered the primary or default one, it is permitted for the representation qualifier for the element or attribute of that type to be left blank.

[Rule DOC2]	Within a GJXDM-conformant reference schema, elements and attributes with the same semantic concept SHALL share the same summary and full description.

Rationale	A summary and a full description describes a semantic concept. Descriptions about the associated type should go into the summary and full description of that type, not in its corresponding elements.

5.7.1. Specific Typing

[Rule ATD1]	GJXDM-conformant schemas SHALL NOT declare attributes or elements to be of type `xsd:anyType` or `xsd:anySimpleType`,

Rationale	In accordance with the principle of avoidance of wildcards, GJXDM schemas should not be able to pass untyped content. All content should have a comprehensible set of values that can be parsed. The type xsd:anyType allows untyped XML content to be carried as a payload. The type xsd:anySimpleType is a union of all possible simple types, and so provides no purposeful constraint on payload content.

5.8. Attribute Declarations

5.8.1. Global Attributes

The GJXDM distribution features attributes that are common to all elements. These common attributes are declared as attribute groups and utilize the following rule.

[Rule ATD2]	If a Schema Expression contains one or more common attributes that apply to all elements contained or included or imported therein, the common attributes SHOULD be declared as part of a global attribute group.

Rationale	For example: see the Global JXDM global attribute group named ”SuperTypeMetadata”

5.8.2. Consistency of Attribute Content

[Rule ATD3]	GJXDM-conformant schemas MUST NOT use the `default` attribute of the `xsd:attribute` element.

Rationale	The default attribute is used in conjunction with optional elements in attribute declarations. It provides a value for the attribute if the attribute does not appear. Such values are yielded to XML instance processing applications after schema validation occurs. The use of this attribute causes data presented to applications to be different than the data that appears in the instances themselves, in violation of the principle of invariant content.

[Rule ATD4]	GJXDM-conformant schemas MUST NOT use the `fixed` attribute of the `xsd:attribute` element, except when used in conjunction with the `use` attribute having the value `"required".`

Rationale	The fixed attribute is used to ensure that a used attribute always has a specific value. When applied to an optional element, it acts like the default attribute, changing the content of the attribute upon schema validation. Using it with required attributes ensures that valid content always has the specific value, while allowing the pre- and post-validated content to be identical.

6. Annotations

All GJXDM-conformant schemas must include documentation. Some documentation is intended to be human readable ("user information"), and other documentation is machine-readable ("application information"). These terms come from **[XMLSchemaStructures]**, a normative source.

6.1. User Information ("documentation") Elements

[Rule DOC3]	Within GJXDM-conformant schemas, the content of `xsd:documentation` elements SHALL NOT contain structured XML data.

Rationale	According to the XSD specification the content of xsd:documentation elements is intended for human consumption. XML content is intended for machine consumption. As such, any XML content appearing in xsd:documentation should be in the context of human-targeted examples, and should be escaped using &lt and &gt.

See **[SchemaForXMLSchema]**, the schema for XML Schema, as an example.

[Rule DOC4]	The attribute `xml:lang` SHALL be used to indicate the language of user information in GJXDM-conformant schemas.

Rationale	XSD spec indicates that user info should use xml:lang to indicate the language of the user info. Note that the value of xml:lang is inherited by child elements, so the attribute need not be owned directly by the xsd:documentation element.

6.2. Application Information ("appinfo") Elements

[Rule DOC5]	An `xsd:appinfo` element SHALL contain well-formed XML data that conforms to **[XMLNamespaces]**.

Rationale	Application information elements are intended for "automatic processing", and so should contain machine-oriented data, XML. Such XML should conform to specifications.[3]

[Rule DOC6]	Any element that is an immediate child of an `xsd:appinfo` elements SHALL be in a namespace.

Rationale	Appinfo may contain XHTML data (which has no schema), or GJXDM appinfo data (which has a schema). Use of default namespace is OK, but content has to have a real namespace. The XML namespaces specification includes the

[3] The XML Schema specification states "{user information} is intended for human consumption, {application information} for automatic processing."

	concept of non-namespaced content. Non-namespaced data confounds the concept of distinctly identifiable data definitions.

[Rule DOC7]	Within a GJXDM-conformant reference schema, a namespace that is a descendent of an `xsd:appinfo` element SHALL be imported using the `xsd:import` element.

Rationale	The import of appinfo content is not strictly required by the XSD specification, but some tools break without it, and it helps users maintain connections between namespaces and implementations.

6.3. Types of Annotations in Reference Schemas

6.3.1. xsd:documentation: Summary

In keeping with **[XMLSchemaDatatypes]**, the content of xsd:documentation elements is intended for human consumption, not machine consumption. As such, it should contain text, not XML, except when the intent is to provide XML examples. In such cases, the escape sequences "<" and ">" should be substituted for the XML brackets "<" and ">" respectively.

[Definition summary documentation element]	
	Within a GJXDM-conformant reference schema, a *summary documentation element* SHALL be defined as an element `xsd:documentation` which does not own an attribute `structures:annotationCategoryURI`.

Rationale	Any documentation element which does not carry an annotationCategoryURI attribute is assumed to be a summary.

[Rule DOC8]	Within a GJXDM conformant reference schema, there SHALL exist a summary documentation element as a child of an element `xsd:annotation` that is a child of every occurrence of the following elements: `xsd:import`, `xsd:simpleType`, `xsd:complexType`, `xsd:group` when acting as a model group definition schema component, `xsd:attributeGroup` when acting as an attribute group definition schema component, `xsd:element` when acting as an element declaration schema component, or `xsd:attribute` when acting as an attribute declaration schema component.

Rationale	These elements are the elements that act as definitions. They must be annotated properly, including basic summaries. Note that the specific "acting" clauses are clearly defined in the XSD specification.

6.3.2. xsd:documentation: Full Description

6.3.3. xsd:appinfo: For Components

6.3.4. xsd:appinfo: List of Abbreviations

7. Subset Schemas

A subset schema is a GJXDM-conformant schema which is derived from a GJXDM-conformant reference schema. The primary rule is that any instance that validates to the subset schema must validate to the reference schema.

Note that these rules are not intended to act as a guide or procedure for generating subset schemas from reference schemas. They are intended to act as a set of constraints that ensure that generated schemas are properly defined subsets.

[Definition GJXDM-conformant subset schema]	A *GJXDM-conformant subset schema* is defined as a GJXDM-conformant schema which is derived from a GJXDM-conformant reference schema according to the rules provided by this document.

Rationale	A subset schema is as defined by this document.

[Rule SSR2]	A GJXDM-conformant subset schema MUST be constructed such that any instance that validates against the subset schema SHALL validate to the reference schema on which it is based. All other rules regarding subset schemas are designed to support this rule.

Rationale	The most important rule regarding subset schemas is that they are to be transparent to the validating application. Any instance that validates to the subset schema must be able to validate against the reference schema. In this way, the subset schema is a schema for documents that contain a subset of the content available to documents that validate against the reference schema.

[Rule SSR3]	A GJXDM-conformant subset schema SHALL be derived only via transformations explicitly allowed by this document.

Rationale	This document describes all of the transformations available to produce subset schemas. Other transformations do not result in valid subsets. If additional transformations are discovered, they should be added to this specification.

[Rule SSR4]	A GJXDM-conformant subset schema SHALL be composed of the content of the GJXDM-conformant reference schema, modified by transformations allowed by this document.

Rationale	Subset schemas are derived from reference schemas. This means that a subset schema operates on the target namespace and content of the reference schema. It may act as a replacement for the reference schema, for certain application processing or human browsing.

When transforming from a reference schema to a subset schema, requirements of outside sources must be maintained. If an element is used by an outside source, then it can't be deleted. If a type uses an element, then that element must be defined. All such requirements must be kept in mind as the subset schema is constructed.

[Rule SSR5]	The derivation of GJXDM-conformant subset schemas is subject to the rules of XML Schema. No permitted transformations obviate this requirement.

Rationale	These rules may specify that an element may be omitted, but that does not override the requirements of XSD. All types, elements, etc., that need to be validated should be included within the subset schema.

Note that using these rules to derive a schema from a valid subset schema will generate a valid subset schema. A valid subset of a valid subset is itself a valid subset.

7.1. Schema Document Element

[Rule SSR6]	The subset schema may omit any of the following child elements of the GJXDM-conformant reference schema's `xsd:schema` document element: `xsd:import, xsd:simpleType, xsd:complexType, xsd:group, xsd:attributeGroup, xsd:element, xsd:attribute`.

Rationale	Many of the definition schema components may be omitted, if they are not otherwise required. They are "omittable." This does not mean that users must remove them, or that it won't be a violation of XSD for them to omit such components. Note that omission of an element implies omission of the element and all child elements, attributes, and namespace prefix definitions.

7.2. Annotations

[Rule SSR7]	Any element `xsd:annotation`, `xsd:appinfo`, or `xsd:documentation` may be omitted from a subset schema.

Rationale	Annotations are merely informative to the XSD validation process, and so may be dropped. Specific annotations may be required in reference schemas, but may be omitted from subset schemas.

7.3. Simple Type Definition

[Rule SSR8]	An attribute `final` owned by the element `xsd:simpleType` may be expanded in scope. It may be set to `"#all"`, or to a superset of its value, or to a valid value if empty.

Rationale	Subclasses may wish to prevent elements from being substituted via element / substitution group substitution. In such a case, the value for final may be expanded to satisfy requirements.

7.3.1. Simple Content Definition

Note that these rules apply to simple types, as well as to simple content in a complex type.

[Rule SSR9]	An element `xsd:enumeration`, child of element `xsd:restriction`, may be omitted, provided that it has a sibling element `xsd:enumeration` which is not omitted. The final `xsd:enumeration` child of an element `xsd:restriction` SHALL NOT be omitted.

Rationale	If the last xsd:enumeration is omitted, it drastically expands the set of legal values for the type.

[Rule SSR10]	The following elements, children of element `xsd:restriction`, may be added or adjusted to reduce the set of legal values: `xsd:minExclusive, xsd:minInclusive, xsd:maxExclusive, xsd:maxInclusive, xsd:minLength, xsd:maxLength.`

[Rule SSR11]	The following elements, children of element `xsd:restriction`, may be added to reduce the set of legal values: `xsd:totalDigits, xsd:fractionDigits, xsd:length, xsd:pattern`.

Rationale	Simple type facets may be added or strengthened to limit the available set of valid values. In no case is it acceptable to enlarge the set of allowable values.

7.4. Complex Type Definition

Note that the rules specified in section 7.3.1, Simple Content Definition, apply to complex type definitions with simple content.

[Rule SSR12]	The attribute `block` owned by element `xsd:complexType`, or by element `xsd:element`, may be expanded in scope. It may be set to `"#all"`, or to a superset of its original value, or to a valid value if empty.

Rationale	Block prevents subtypes from being substituted for the specified element. This may be enabled or strengthened.

[Rule SSR13]	The attribute `final` owned by element `xsd:complexType` may be expanded in scope. It may be set to `"#all"`, or to a superset of its value, or to a valid value if empty.

Rationale	Final prevents substitution groups from being used in element substitution. This may be enabled or strengthened

7.4.1. Attribute Declarations

[Rule SSR14]	The element `xsd:attribute`, when used as an attribute use schema component, may be omitted, if the value of its attribute use is "`optional`" or "`prohibited`"

Rationale	Attributes which are optional may be removed.

Note that prohibited attributes may be omitted, but that does not imply that types derived from a type with a removed prohibited attribute may add the prohibited attribute. Schemas must be built from the reference schemas, and then subset. They should not be built from the subset schemas, at the risk of invalidity or non-conformance.

[Rule SSR15]	The attribute `xsd:attribute,` when used as an attribute use schema component, MAY have a modified value which narrows the use of the attribute. If the reference value is "`optional,`" then the subset may have any value. Otherwise, it MUST have the original value.

Rationale	Optional attributes may be required or removed. Those attributes which are already required or prohibited must stay that way.

[Rule SSR16]	An element `xsd:attributeGroup` which does not act as an attribute group definition may be omitted only if all components declared by the attribute group are omittable.

Rationale	An attribute group may only be removed if all of its components are themselves removable. If any component of the attribute group is required, the attribute group must persist.

7.4.2. Complex Content

These rules provide methods for simplifying and reducing the model group defined in complex content.

[Rule SSR17]	An element `xsd:group, xsd:choice, or xsd:sequence`, SHALL NOT be omitted in a subset schema if its reference definition parent element is `xsd:complexType, xsd:extension,` or `xsd:restriction.`

Rationale	group, choice, and sequence that are roots of the particle schema component may not be eliminated, as it has substantial changes on the contents allowable by the schema construct defined by the parent element.

[Rule SSR18]	The element `xsd:group`, when used as a particle schema component, may be omitted from the subset schema, only if its reference element has a `minOccurs` attribute with a value of "`0`", or if all components declared by the group are themselves omittable.

Rationale	A group may be removed if it is, as a whole, optional.

[Rule SSR19]	The element `xsd:choice` may be omitted from the subset schema only if its reference element has a `minOccurs` attribute with a value of `"0"`, or if all components declared by the choice model group are themselves omittable.

Rationale	A choice element may be removed if it is, as a whole, optional

[Rule SSR20]	The element `xsd:sequence` may be omitted from the subset schema only if its reference element has a minOccurs attribute with a value of `"0"`, or if all components declared by the sequence model group are themselves omittable.

Rationale	A sequence may be removed if it is, as a whole, optional.

[Rule SSR21]	For any of `xsd:group` when used as a particle schema component, `xsd:element` when used as a particle schema component, `xsd:choice`, or `xsd:sequence`, the attributes `minOccurs` and `maxOccurs` may be adjusted to narrow the occurrence of subcomponents.

Rationale	A group may be removed if it is, as a whole, optional. Note that the "particle schema component" language is provided by **[XMLSchemaStructures]**.

[Rule SSR22]	The element `xsd:element` when used as a particle schema component may be omitted from the subset schema only if its reference element as a `minOccurs` attribute with a value of `"0"`.

Rationale	A group may be removed if it is, as a whole, optional.

7.5. Element Definition

[Rule SSR23]	The attribute `final` of element `xsd:element` may be expanded in scope. It may be set to "`#all`", or to a superset of its value, or to a valid value if empty.

Rationale	Final prevents substitution groups from being used in element substitution. This may be enabled or strengthened

[Rule SSR24]	The attribute `block` of element `xsd:element` may be expanded in scope. It may be set to "`#all`", or to a superset of its value, or to a valid value if empty.

Rationale	Block prevents subtypes from being substituted for the specified element. This may be enabled or strengthened.

[Rule SSR25]	The attribute `nillable` on an element `xsd:element` may be set to `"false"` regardless of the value of `nillable` in the reference element.

8. Constraint Schemas

[Definition GJXDOM-compatible constraint schema]	A *GJXDM-compatible constraint schema* is a schema that follows the rules for GJXDM-compatible constraint schemas as specified by this document.

Rationale	Definition of term.

[Rule CSR1]	A GJXDM-compatible constraint schema has a `targetNamespace` identical to the `targetNamespace` of a GJXDM-conformant reference schema.

Rationale	Constraint schemas operate by adding additional validation testing on already-valid content. Content must validate against GJXDM-conformant reference schemas to be considered a GJXDM-conformant instance.

9. XML Instance Rules

This specification attempts to restrict XML instance data as little as possible, while still maintaining interoperability.

[Definition GJXDM-conformant instance]
A *GJXDM-conformant instance* is a document or data set that satisfies the rules for GJXDM-conformant instances as specified in this document.

Rationale	XML data may be referred to as a GJXDM-conformant instance if it conforms to this specification

[Rule IND1]	A GJXDM-conformant instance MUST have a document element that is defined in a GJXDM-conformant schema.

Rationale	The root of a GJXDM-conformant instance MUST be an element defined in a GJXDM-conformant schema. The term *document element* is defined by **[XMLInfoSet]**.

[Rule IND2]	A GJXDM-conformant instance MUST validate to the reference schemas for namespaces contained in the instance, and for namespaces required for validation.

Rationale	Reference schemas determine the exchange language. Derived schemas, and subsets, are for specific applications, but it is the reference schemas that set the standard for conformance.

GJXDM embraces the use of XML schema instance attributes, including xsi:type, xsi:nil, and xsi:schemaLocation, as specified by **[XMLSchemaStructures]**.

[Rule IND3]	Within a GJXDM-conformant instance, the meaning of an element with no content is undefined. There SHALL NOT be a meaning assigned to an element with no content.

Rationale	Elements without content have no specific meaning within GJXDM. The lack of data should not be interpreted to mean anything other than that such data is not present.

The GJXDM does not require a specific encoding, or specific requirements for the XML prolog, except as specified by **[XML]**.

Appendix A Supporting Files

A.1 Schema for Structures Namespace

```
<?xml version="1.0" encoding="UTF-8"?>
<schema
    attributeFormDefault="qualified"
    targetNamespace='http://www.it.ojp.gov/jxdm/structures/1'
    xmlns:this='http://www.it.ojp.gov/jxdm/structures/1'
    xmlns='http://www.w3.org/2001/XMLSchema'>

  <import
      namespace="http://www.w3.org/XML/1998/namespace"
      schemaLocation="xml.xsd"/>

  <attribute name="sequenceID" type="integer">

  <complexType name="ReferenceType" final="true" block="true">
    <attribute name="reference" type="IDREF" use="required"/>
    <attribute ref="xml:id" use="optional"/>
  </complexType>

  <element name="Relationship">
    <complexTypefinal="true" block="true">
      <attribute name="relationshipURI" type="anyURI"
          use="required"/>
      <attribute name="relationshipObject" type="IDREF"
          use="required"/>
      <attribute name="relationshipSubject" type="IDREF"
          use="required"/>
      <attribute ref="xml:id" use="optional"/>
    </complexType>
  </element>

</schema>
```

A.2 Schema for entity appinfo namespace

```
<?xml version="1.0" encoding="UTF-8"?>
<schema
    targetNamespace="http://www.it.ojp.gov/jxdm/appinfo/2"
    version="1.0"
    xmlns:this="http://www.it.ojp.gov/jxdm/appinfo/2"
    xmlns="http://www.w3.org/2001/XMLSchema">

  <element name="info">
    <complexType>
      <sequence>
        <element form="qualified" maxOccurs="unbounded"
            minOccurs="0" name="base">
          <complexType>
            <attribute form="qualified" name="namespace"
                type="anyURI" use="required"/>
            <attribute form="qualified" name="name"
                type="NCName" use="required"/>
          </complexType>
        </element>
```

```
      </sequence>
      <attribute form="qualified" name="deprecated"
          type="boolean" use="required"/>
    </complexType>
  </element>

</schema>
```

A.3 Schema for xml namespace

```
<?xml version="1.0" encoding="UTF-8"?>
<schema
    targetNamespace="http://www.w3.org/XML/1998/namespace"
    xmlns="http://www.w3.org/2001/XMLSchema">

  <attribute name="id" type="ID"/>

</schema>
```

Appendix B Normative Abbreviations

This is a table of normative abbreviations, acronyms, and word truncations to be used as specified in [Rule GNR6].

Term	Definition
ID	Identifier
ORI	Orion value

Appendix C References

[Global] http://it.ojp.gov/global

[OED] *Oxford English Dictionary*, Second Edition, 1989. Available at http://dictionary.oed.com/

[RDFConcepts] http://www.w3.org/TR/2004/REC-rdf-concepts-20040210/

RDF data model http://www.w3.org/TR/2004/REC-rdf-concepts-20040210/#section-data-model

[RFC2119] S. Bradner, Key words for use in RFCs to Indicate Requirement Levels, http://www.ietf.org/rfc/rfc2119.txt, IETF RFC 2119, March 1997.

[RFC3986] Berners-Lee, T., et al: Uniform Resource Identifier (URI): Generic Syntax, Request for Comments 3986, January 2005 available from http://www.ietf.org/rfc/rfc3986.txt

[SchemaForXMLSchema] The schema for XML Schema is available at http://www.w3.org/2001/XMLSchema.xsd

[XML] Extensible Markup Language (XML) 1.0 (Third Edition), W3C Recommendation 04 February 2004, available at http://www.w3.org/TR/2004/REC-xml-20040204/

EBNF Notation http://www.w3.org/TR/2004/REC-xml-20040204/#sec-notation

IDREF constraint http://www.w3.org/TR/2004/REC-xml-20040204/#idref

[XML-ID] xml:id Version 1.0, W3C Proposed Recommendation 12 July 2005, available from http://www.w3.org/TR/2005/PR-xml-id-20050712/.

[XMLInfoSet] XML Information Set (Second Edition), W3C Recommendation 4 February 2004. Available from http://www.w3.org/TR/2004/REC-xml-infoset-20040204/

[XMLNamespaces] Namespaces in XML, World Wide Web Consortium 14-January-1999, available at http://www.w3.org/TR/1999/REC-xml-names-19990114/

NCName http://www.w3.org/TR/REC-xml-names/#NT-NCName

[XMLNamespacesErrata] Namespaces in XML Errata, 6 December 2002, available from http://www.w3.org/XML/xml-names-19990114-errata

[XMLSchemaDatatypes] XML Schema Part 2: Datatypes Second Edition, W3C Recommendation 28 October 2004, available at http://www.w3.org/TR/2004/REC-xmlschema-2-20041028/

[XMLSchemaStructures] XML Schema Part 1: Structures Second Edition, W3C Recommendation 28 October 2004, available at http://www.w3.org/TR/2004/REC-xmlschema-1-20041028/

Appendix D Revision History

Revision	Date	Modifications
0.4	2005-08-23	Removed in-document tasks. Formatted for public review.
0.3	2005-08-10	Processed comments by XSTF
0.2	2005-07-21	Remove email addresses. Inserted Appendix B for acronyms.
0.1	2005-07-21	Initial draft by Webb Roberts

Appendix E Glossary

This glossary is informative only. There is no normative content. All normative definitions appear within.

GJXDM

Global Justice XML Data Model

GJXDM-conformant reference schema

A schema that acts as the definition for its namespace. It maintains documentation that allows it to be shared and interoperable as a complete GJXDM component.

GJXDM-conformant schema

A schema that maintains the XML Schema syntax requirements of GJXDM, while not necessarily containing all content for a namespace, and not necessarily containing all documentation needed for full interoperability and GJXDM integration. GJXDM-conformant reference schemas and GJXDM-conformant subset schemas fall under this category, as do extension schemas and document schemas.

GJXDM-conformant subset schema

A schema, based on a GJXDM-conformant reference schema that is built to validate a subset of the content of the full reference schema. It is built from a reference schema using rules specified in this document.

GJXDM constraint schema

A schema, used in conjunction with GJXDM-conformant schema, that applies a set of user-designated constraints on XML data instances.

Global

The Global Justice Information Sharing Initiative. For more information, see **[Global]**

XSTF

The Global XML Structure Task Force, the organization supervising the GJXDM

Appendix F Notices

This document and the information contained herein is provided on an “AS IS” basis and the authors DISCLAIM ALL WARRANTIES, EXPRESS OR IMPLIED, INCLUDING BUT NOT LIMITED TO ANY WARRANTY THAT THE USE OF THE INFORMATION HEREIN WILL NOT INFRINGE ANY RIGHTS OR ANY IMPLIED WARRANTIES OF MERCHANTABILITY OR FITNESS FOR A PARTICULAR PURPOSE.